Engineering Coordination
A Methodology for the Coordination of Planning Systems

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim Fachbereich Informatik und Mathematik
der Johann Wolfgang Goethe – Universität
in Frankfurt am Main

von
René Schumann
aus Delmenhorst

Frankfurt (2010)
(D 30)

vom Fachbereich Informatik und Mathematik der

Johann Wolfgang Goethe – Universität als Dissertation angenommen

Dekan: Prof. Dr. Tobias Weth ....

Gutachter: Prof. Dr. Ingo J. Timm, ...
Prof. Dr. Jürgen Sauer, ...
Prof. Dr. Ulrich Schwanecke ....

Datum der Disputation: 06.10.2010 ....

Acknowledgement

The work this thesis is based on, has been done over several years and at three different locations. During my appointment at the OFFIS Institute for Information Technology in Oldenburg, while I was employed at the Goethe University in Frankfurt am Main, and during my research stays at the Artificial Intelligence group at the Rey Juan Carlos University in Móstoles (Spain).

The issue of coordination of autonomous planning systems has been at the center of my research for more than six years now. It started when I was a student at the Carl von Ossietzky University Oldenburg. This project took longer than I expected and I moved to locations I never thought I would visit when I started it. Consequently, over the course of time I have to thank a number of people I have meet during this time at the various locations.

First of all, I would like to thank my main supervisor Prof. Dr. Ingo J. Timm (Goethe University Frankfurt am Main). I met him in 2005 at a conference and moved to Frankfurt in 2007 to become part of his group. I would like to thank him for the opportunity to move to Frankfurt. It was not only a geographical change, but also offered me a complete new perspective on research. I am very grateful for all the valuable advice and encouragement that he still gives me.

Furthermore, I would like thank apl. Prof. Dr. Jürgen Sauer (Carl von Ossietzky University Oldenburg). He supervised my diploma thesis in 2004 and to this day still gives me valuable advice. He encouraged me to start with my PhD and support me continuously even after I left Oldenburg.

I also want to thank Prof. Dr. Ulrich Schwanecke (Hochschule RheinMain University of Applied Sciences) for the valuable advice he gave me since I joined the Sensyble graduate school.

I am deeply grateful for the hospitality and valuable discussions of Prof. Dr. Sascha Ossowski and his entire group. I spent 10 days in June 2009 and three month (Feb.–Apr.) in 2010 as a visiting researcher in his group. Large parts of this thesis have been written there.

I want to express my gratitude to Dr. Christoph Mayer (OFFIS Institute for Information Technology). Christoph Mayer was the division manager of the R&D division Business Information Management at the OFFIS Institute, and my first superior. I would like to thank him for his support, his advice and the chance to learn a lot about projects and leadership in science.

A very special thanks goes to the entire Information Systems and Simulation group at the Goethe University Frankfurt am Main. The people that accompanied me during my time at this group (in addition to Prof. Dr. Ingo J. Timm) are Marion Terrell, Dr. Andreas Lattner, Tjorben Bogon, Yann Lorion, Pascal Katzenbach and Jörg Dallmeyer. I have to thank all of them, and in particular Dr. Andreas Lattner, for their support and advices.

Also, a very special thank you goes to my dear friend Mathias Uslar (OFFIS Institute for Information Technology), a former colleague of mine. Especially in my first years of my endeavors in research he supported me with a lot of valuable advice while in the last years he offered much needed moral support.

I also have to thank Jens Oehlerking (University Oldenburg), who also supported me during the writing of this thesis.

Another person I met during my time as PhD student, who became a friend that also helped me is Dr. Leif Meier (Proctor & Gamble Service GmbH). He got me involved in the container terminal management problem and we have worked collaboratively on this issue. Also, I would like to thank him for the planning systems he provided to me.

I was also supported by Dr. David Sabel (Goethe Universität Frankfurt) and want to thank him for supporting the Latex template this thesis uses.

I would like to thank Dr. Michael Schwind (Goethe Universität Frankfurt) for the insightful discussions concerning combinatorial auctions.

I am very grateful to my former diploma students, who have contributed to aspects of my research that can partially be found in this thesis. In particular, I have to mention Thomas Timmermann (Ernst & Young GmbH) and Zijad Kurtanovic (Healy Hudson GmbH).

Also, I have to thank the students who participated in the practical course “Praktikum Wirtschaftsinformatik und Simulation” in the winter term 2009/2010 for their efforts in developing planning systems and encapsulate them using web services.

While I was employed at the Goethe University I was additionally supported by GRADE, the Goethe Graduate Academy (former Otto Stern School) supporting me with subsidies for childcare. I received travel funding from the Herrmann Willkomm-Stiftung in 2008 and 2009. Furthermore, I was supported by the EU-COST Action Agreement Technologies. I received a grant for a short-term scientific mission (STSM) for a 10 day visit of the group of Prof. Dr. Sascha Ossowski in 2009. Finally, I was supported by the German Academic Exchange Service (DAAD) with a short-term grant for doctorate students (grant number D/09/48031). This scholarship allowed me to stay in Spain for three month in 2010.

For their ongoing support in so many different ways I would like to thank my parents. I am very grateful for their support.

Especially in the last months of this thesis, my family supported me very much. They had to put up with my stress and the fact that I was not there for three month. Therefore, I am very grateful.

I would like to thank my two sons Philipp Ephraim and Simon Casimir for their ongoing lecture they give to me to focus on the important aspects of life. They were born during the time this thesis has been written, and know their father only as a PhD. student, so far.

All these persons have influenced me, and consequently, the way this thesis looks like. But there would be no thesis in any way without the support and love of my wife. I am deeply indebted to Jennifer for the advice and support she provided to me during all this time. She brought me back on track when I lost sight of reality.

Zusammenfassung

Reale Planungsprobleme, wie etwa die Produktionsplanung in einer Supply Chain, sind komplex Planungsprobleme. Eine übliche Strategie derart komplexen Problemen zu lösen, ist es diese Probleme in einfachere Teilprobleme zu zerlegen und diese dann separat, meist sequentiell, zu lösen (divide-and-conquer Strategie). Dieser Ansatz erlaubt die Erstellung von (suboptimalen) Plänen für eine Reihe von realen Anwendungen und ist heute in den Organisationsstrukturen von größeren Unternehmen institutionalisiert worden. Allerdings werden Abhängigkeiten zwischen den Teilproblemen nicht ausreichend berücksichtigt, da die Partialprobleme sequentiell ohne Feedback gelöst werden. Die erstellten Teillösungen müssen deswegen oft nachträglich koordiniert werden. Die Beispiele, die in dieser Arbeit genutzt werden, sind die Koordination von Planungssysteme in der Produktion und Distribution von Gütern, die Koordination der Planungssysteme von Partner eines Supply Chain, und die Koordination der Planungssysteme die im Rahmen des Container Terminal Managements.

In dieser Arbeit wird die operative Koordination von existierenden Planungssystemen betrachtet. Dabei wird von einer inhärent gegebenen Problemstellung ausgegangen in der für verteilte, abhängige Planungsentscheidungen automatische Planungssysteme eingesetzt werden. Die Pläne dieser Planungssysteme müssen koordiniert werden, um die Realisierbarkeit dieser Pläne unter Berücksichtigung evtl. wechselseitiger Abhängigkeiten der Planungsprobleme zu gewährleisten. Je nach gegebener Problemstellung kann dabei der Fokus neben der reinen gemeinsamen/parallelen Realisierbarkeit einer Menge von verteilt erstellten Plänen auch die Verbesserung einer Gesamtsystemperformance im Fokus liegen. Eine Koordination von Planungsprozessen auf der taktischen oder strategischen Ebene wird in dieser Arbeit nicht betrachtet. Ein Beispiel für derartige taktische oder strategische Koordinationsprobleme sind etwa die Auswahl eines Netzwerkes oder die Zerlegung des Gesamtproblems in unterschiedliche Teilprobleme.

Im Rahmen der hier vorliegenden Arbeit wird zuerst ein Einblick in die Grundlagen von Planungssystemen und Systemen der verteilten Künstlichen Intelligenz gegeben. Anschließend wird im dritten Kapitel ein breiter Überblick über den Stand der Forschung gegeben. Dabei werden mehrere Bereiche untersucht. Zuerst wird die Modellierung von Abhängigkeiten betrachtet und dabei grundsätzliche

Überlegungen zu Problemen und Komplexität bei der Koordination von autonomen Planungssystemen vorgestellt.

Anschließend wird das Gebiet der Koordination wird in verschiedenen Forschungsgebieten, wie etwa der verteilten Künstlichen Intelligenz, den Wirtschaftswissenschaften oder der Spieltheorie untersucht. Weiterhin wird das Gebiet des agentenorientiertem Software Engineering betrachtet. Dieses wird insbesondere auf dessen Beitrag zur Wiederverwendung von bereits entwickelten Methoden hin betrachtet. Dabei ist festzuhalten, dass der Hauptbeitrag der Wiederverwendung von Konzepten des Gebietes des agentenorientierten Software Engineering im Rahmen von Agentenframeworks stattfindet. Darüber hinaus bietet der aktuelle Stand der Forschung kaum Möglichkeiten bzw. Methoden, die die Wiederverwendung von existierenden Verfahren adressieren. Als Ausnahmen sind lediglich die Ansätze von Jonker et al. [JTY05] für die Nutzung von Organisationsformen und Bussmann et al. [BJW03] für die Auswahl von Interaktionsprotokollen zu nennen. Beide Ansätze basieren auf der Idee existierende Verfahren/Methoden anhand von spezifischen Kriterien zu klassifizieren und über diese Spezifikation der Charakteristika der aktuellen Situation eine gerichtete Suche in den existierenden Verfahren nach diesen Kriterien durchzuführen.

In dieser Arbeit werden aus diesem Grund die existierenden Verfahren zur Klassifikation von Koordinationsverfahren untersucht. Existierende Klassifikationen orientieren sich an eingesetzten Technologien (etwa bei Stockheim et al. [SSWG02]), der Perspektive auf das System (etwa bei Schumacher [Sch01]) oder auf Basis von allgemeinen Theorien über Koordination (etwa bei Busi et al. [BCGZ01]). Diese Klassifikationskriterien spiegeln die unterschiedlichen Forschungsrichtungen und Theorien wieder, die einen Beitrag zur Erforschung der Koordination im Rahmen der verteilten Künstlichen Intelligenz geleistet haben. Keine dieser Klassifikationsansätze erlaubt eine Aussage über Einsatzmöglichkeiten und Designentscheidungen der jeweiligen Koordinationsverfahren aus. Aus diesem Grund wird im Rahmen dieser Arbeit einen problemorientierten Ansatz vorgestellt. Durch die Identifikation von Charakteristika von Koordinationsproblemen und der Herausarbeitung dieser Anforderungen an ein Koordinationsverfahren wird das Konzept der Coordination Requirements entwickelt. Dies sind formalisierte Anforderungen an ein Koordinationsverfahren, die das Koordinationsverfahren erfüllen muss, um in der gegeben Situation anwendbar zu sein. Es wird somit die Möglichkeit eröffnet eine formale Spezifikation des Sachverhalts der Anwendbarkeit für eine Koordinationsaufgabe zu spezifizieren.

Im Abschnitt 4.1 präsentieren wir auf Basis der in dieser Arbeit verwendeten beispielhaften Koordinationsprobleme die folgenden sechs Charakteristika:

- Ist ein Allokationsproblem vorhanden?

- Sind die lokalen Zielfunktionen vergleichbar?
- Sind die Planungssysteme homogen?
- Existiert eine globale Zielfunktion?
- Ist information hiding notwendig?
- Existieren zyklische Abhängigkeiten?

Auf Basis dieser sechs Charakteristika werden die vorgestellten Koordinationsverfahren klassifiziert. Dies alleine ist zwar nicht ausreichend für die Identifikation von vorhanden geeigneten Koordinationsverfahren, allerdings kann es, wie in den durchgeführten Fallstudien gezeigt wird, den Auswahlprozess beschleunigen, da schnell geeignete Kandidaten identifiziert werden können, die in einer genaueren Analyse auf deren Anwendbarkeit hin untersucht werden können.

Für die strukturierte Durchführung der Identifikation von geeigneten Koordinationsverfahren wird im Rahmen dieser Arbeit ein Prozessmodell vorgestellt, der sogenannte ECo-Prozess (Engineering Coordination). Dieser Prozess soll sowohl den Auswahlprozess, als auch, sofern nötig, den Designprozess eines neuen Koordinationsverfahrens unterstützen. Der Prozess ist in die folgenden Schritte eingeteilt, die iterativ durchlaufen werden können:

- Modellierung der Problemstellung und des relevante Kontextes
- Formulierung von Anforderungen an einen Koordinationsmechanismus
- Auswahl/Entwurf eines Koordinationsmechanismuses
- Implementierung des Koordinationsverfahrens
- Evaluation des Koordinationsverfahrens

Diese Schritte werden im Rahmen der vorliegenden Arbeit detailliert beschrieben. Die Modellierung der Problemstellung stellt dabei den ersten Schritt dar, um die Problemstellung analytisch zugänglich zu machen.

Wie bereits oben erläutert ist die Aufgabe der Coordination Requirements den Sachverhalt der Anwendbarkeit zu operationalisieren. Die Anforderungen sollen dabei auf Basis der vorangegangen Modellierung formuliert werden, d.h. die Requirements haben eine formale Grundlage. Die in Frage kommenden Koordinationsverfahren sollen im nächsten Schritt, der Auswahl eines geeigneten Koordinationsverfahrens, daraufhin betrachtet werden, ob sie die aufgestellten Anforderungen erfüllen. Dies wird als qualitative Evaluation bezeichnet. Da die Coordination Requirements eine formale Definition besitzen, kann diese Evaluation

ebenfalls formal durchgeführt werden. Das Ergebnis der qualitativen Evaluation ist eine Menge von möglichen Koordinationsverfahren, die im gegeben Kontext anwendbar sind. Dies erlaubt allerdings keine weiteren Aussagen, wie gut, im Sinne eines global erreichbaren Qualitätsmaßes, diese Verfahren sind. Dies geschieht erst in der später folgenden quantitativen Evaluation. Der von den Anforderungen getrieben Auswahlprozess ist ein Kernstück der hier vorgestellten Arbeit. Durch die Formulierung der Anforderungen und der Annotation eines Koordinationsmechanismus bezüglich der erfüllten und nicht erfüllten Anforderungen werden die Motive für Designentscheidungen dieses Verfahren expliziert. Wenn Koordinationsverfahren anhand dieser Anforderungen klassifiziert werden können, ist es weiterhin möglich den Auswahlprozess zu vereinfachen und zu beschleunigen. Stellt sich in diesem Schritt heraus, dass kein existierenden/bekannter Koordinationsansatz alle Anforderungen erfüllt ist ein solches Koordinationsverfahren nun auf Basis des Modells des Anwendungsfalles und der aufgestellten Coordination Requirements zu entwerfen.

Um eine quantitative Evaluation durchführen zu können, ist es notwendig die in Frage kommenden Verfahren zu implementieren. Für die Unterstützung der Implementierung eines Koordinationsansatzes wird in dieser Arbeit zusätzlich der CoPS Prozess (Coordination of Planning Systems) vorgeschlagen. Der CoPS Prozess erlaubt einen ganzheitlichen systematischen Ansatz für den Entwurf und die Implementierung eines Koordinationsverfahrens. Dabei werden im CoPS sowohl Aspekte auf der globalen Netzwerkebene, wie die Auswahl von Konversationsprotokollen, als auch Entscheidungen, die lokal bei dem Entwurf eines jeden Agenten getroffen werden müssen berücksichtigt. Beispiele für lokale Designentscheidungen sind etwa die Spezifikation von Verhaltensmustern in Konversationen oder die Adaption existierender Planungssysteme, etwa mittels Web Services.

Zur Unterstützung des CoPS Prozesses wird in dieser Arbeit das CoPS Framework vorgestellt. Auf Basis dieses Frameworks soll eine schnellere Implementierung ermöglicht werden. Hierzu bietet das CoPS Framework etwa die Möglichkeit an Konversationsprotokolle als textuelle Beschreibung zu spezifizieren und diese dann automatisiert zum Einen in Verhaltensautomaten zu transformieren, in denen ein lokales Entscheidungsverfahren modelliert werden kann, und zum Anderen kann diese textuelle Beschreibung in Sequenzdiagramme transformiert werden, um den Entwicklern des Systems eine geeignete Dokumentation zu liefern. Ziel des CoPS Frameworks ist es eine Plattform mit Basisfunktionalität eines Agenten bereit zu stellten der für die Koordination von Planungssystemen verantwortlich ist. Dabei ist es die Zielstellung des CoPS Frameworks die Erstellung eines Prototyps eines agentenbasierten Koordinationsmechanismus für existierende Planungssysteme. Dieser Prototyp soll dazu dienen den letzten

Schritt, die Evaluation des Koordinationsverfahrens auf Basis quantitativer Daten zu ermöglichen. Das CoPS Framework ist zurzeit für der produktiven Betrieb nicht geeignet, da hierzu relevante Funktionalitäten, etwa bzgl. der Skalierbarkeit, der Performance und der Persistenz nicht ausreichend unterstützt werden.

Als abschließender Schritt des ECo Prozesses steht die Evaluation. Hierbei soll eine Evaluation der Performance, etwa in Bezug auf die Performance des Gesamtsystems hin, durchgeführt werden. Dieser Schritt wird auch als quantitative Evaluation bezeichnet, um dies zur oben erwähnten qualitativen Evaluation abzugrenzen. Die quantitative Evaluation ist ein wichtiger Prozessschritt, da sie die Überprüfung der beabsichtigten Performancekriterien sicherstellt und somit überprüft wird, ob das koordinierte Gesamtsystemverhalten im akzeptablen Bereich liegt. Hierbei sind etwa Vergleiche mit alternativen Methoden oder eine Einordnung der Lösungsqualität möglich, etwa durch den Vergleich mit Verfahren, die die Coordination Requirments nicht einhalten, und so in der Regel in der Lage sind bessere Lösungen zu finden. Beispielhaft ist hier etwa die zentrale Planung zu nennen, die oft bessere Ergebnisse erzielen kann, als eine Menge von koordinierten verteilten Plänen. Aus Basis der Ergebnisse der quantitativen Evaluation kann dann eine Auswahl eines Koordinationsvefahrens für die gegebene Problemstellung erfolgen.

Neben der reinen Definition eines Prozesses ist es weiterhin notwendig die Anwendbarkeit des Prozesses zu belegen. Des Weiteren ist es für eine Verwendung eines derartigen Prozesses im industriellen Kontext notwendig eine empirische Studie über die Potentiale des Prozesses durchzuführen. Eine derartige soziologisch fundierte empirische Studie ist nicht im Fokus dieser Arbeit. Um die Anwendbarkeit und Nutzbarkeit des ECo-CoPS Ansatzes zu zeigen werden in dieser Arbeit zwei Fallstudien untersucht. Das Ziel dieser Fallstudien ist es die Möglichkeiten dieses Ansatzes zu demonstrieren und auf Grund der ersten Anwendungen des Prozesses Rückschlüsse auf dessen Stärken und weitere Forschungsfragen zu ziehen. Als Fallstudie dient zum einen die Produktion und Distribution von Gütern in einem Unternehmen, bestehend aus drei Planungssystemen zum Scheduling, Packen und für die Tourenplanung. Das zweite Fallbeispiel stellt einen vereinfachten Ausschnitt aus der operativen Koordination einer Supply Chain dar, indem die Schedulingssysteme zweier Unternehmen koordiniert werden müssen. In beiden Fallstudien hat sich der ECo-CoPS Ansatz als sehr hilfreich erwiesen ein strukturiertes, ingenieursmäßiges Vorgehen bei der Identifikation/Implementierung eines Koordinationsverfahrens für autonome Planungssysteme zu ermöglichen.

Im Anschluss an die beiden Fallstudien wird eine kritische Würdigung des ECo-CoPS Ansatzes auf Basis der in den Fallstudien gesammelten Erfahrungen

durchgeführt. Hieran schließt sich eine Zusammenfassung der gesamten Arbeit und ein Ausblick auf sich aus dieser Arbeit ergebenden weiteren Forschungsfragen.

Contents

List of Figures

List of Tables

Nomenclature

ABACO	agent-based coordination
ACL	agent communication languages
ADL	Action Description Language
AI	Artificial Intelligence
AOSE	agent-oriented software engineering
APS	advanced planning systems
ARB	agent resource broker
AUML	Agent UML
BAP	berth allocation problem
CA	coordination agent
COP	constraint optimization problem
CoPS	Coordination of Planning Systems
COTS	commercial off-the-shelf
CSP	cranes scheduling problem
CTM	container terminal management
DAI	Distributed Artificial Intelligence
DCOP	distributed constraint optimization
DCSP	distributed constraint satisfaction problem
ECo	Engineering Coordination
edd	earliest due date first
EMF	Eclipse Modeling Framework

FIPA	Foundation of Intelligent Physical Agents
GPGP	Generalized Partial Global Planning
KIF	knowledge interchange format
KQML	knowledge query and manipulation language
MAS	Multiagent System
MDD	model-driven development
MPL	Mathematical Programming Language
NFA	non-finite automata
OCL	object constraint language
OWL	web ontology language
PAA	planning authority agent
PDA	production data acquisition
PDDL	planning domain definition language
PGP	Partial Global Planning
PIM	platform independent model
pit	point in time
SCM	Supply Chain Management
SLP	storage location problem
SPT	scheduling, packing, and transportation
STRIPS	Stanford Research Institute Problem Solver
TPP	transportation planning problem
TÆMS	Task Analysis, Environment Modeling, and Simulation
VDP	vehicle dispatching problem
VRP	vehicle routing problem

1. Introduction

1.1. Motivation and goal of this study

Real scheduling problems, like production scheduling are very complex. A common strategy to handle such complex problems is the 'divide and conquer' strategy. Thus, the overall problem is divided into simpler sub-problems and these are solved in most cases sequentially. This approach enables the generation of (sub-optimal) plans for a number of real-world-applications and has been implemented in organization structures of companies. Consequently, these partial solutions have to be coordinated. In the research of Distributed Artificial Intelligence (DAI), business administration, and game theory, various coordination mechanisms have been presented. But there is no commonly applicable coordination mechanism [Dec95, p. 2]. Thus, the designer of a system has to choose a coordination mechanism well suited for the situation at hand. Unfortunately, there exists no appropriate and general methodology to select or design a coordination mechanism for a given situation. As the ability to coordinate its activities "with others constitutes a centrepiece of agenthood" [Oss08, p. 2] a huge number of coordination approaches have been presented in the last decades.

To support this claim with facts we outline here some number of Google Scholar search results. Google Scholar is a search engine for scholarly literature[1].

For the search string "`agent coordination`" the search without references but at least a summary returns a result of about 819.000 documents[2]. As the search engine is not specific for a given research discipline it may be useful to restrict the search more towards the field of DAI and multiagent systems, which is primarily addressed in this study. Therefore, we can narrow down the search by adding the search term multiagent system. So the search string is "`agent coordination multiagent`" the result set has about 26.900 elements[3]. If we use the search string "`agent coordination multi agent`" the result set has a size of about 198.000

[1]`http://scholar.google.com/intl/en/scholar/about.html`, Accessed: 04/25/2010

[2]`http://scholar.google.com/scholar?hl=de&q=agent+coordination&lr=&as_ylo=&as_vis=1`, Accessed: 04/24/2010

[3]`http://scholar.google.com/scholar?hl=de&q=agent+coordination+multiagent&btnG=Suche&lr=&as_ylo=&as_vis=1`, Accessed: 04/25/2010

documents[4].

So even if we can reduce the result set by adding more specific terms that may characterize the situation at hand more precisely one has to state that the body of existing research, and therefore the body of proposed coordination techniques as well, is quite large. Moreover, as we point out in this thesis, there exists currently no methodology for selecting appropriate coordination mechanisms. Thus, the identification of appropriate coordination mechanisms is a complex task, for that no structure has been proposed, up to now.

A *coordination mechanism* "determines its [the agents] internal dynamics, i.e., the interactions between agents, as well as the external properties of the society" [Oss98, p. 48]. It is possible within a coordination mechanism to destinction between cases, and apply different *coordination techniques* for different cases. Examples for such case-by-case analysis are the destinctions between predicitive and reactive planning steps, which can be found, for instance, in the ABACO coordination approach [SS05]. Coordination techniques are means for coordination that define interaction protocols or more abstract agents interactions, like auctions, for instance. Coordination techniques do not require any destinction of cases as they focus on the pure coordination and not the context of is applicatibility. The destinction between the terms mechanism and techniques is not common sense in the literature, but to enables a clear presentation.

Within this thesis a process (the ECo process) is presented that clarifies how coordination mechanisms can be chosen and that can assist if no appropriate coordination mechanism is available. ECo is the acronym for *Engineering Coordination*. The ECo process comprises of the following five steps:

- Modeling the coordination environment
- Formulating the coordination requirements
- Choosing/designing coordination mechanisms
- Implementing the coordination mechanisms
- Evaluating the coordination mechanisms

Within this study these steps are detailed out and applied to the coordination process that is necessary for the coordination of our planning systems.

The domain is formally modeled, and based upon this model, coordination requirements are specified. The main idea for the selection of coordination mechanisms is that the coordination mechanisms have to regard the current context and

[4]`http://scholar.google.com/scholar?hl=de&as_sdt=2000&as_vis=1&q=agent+coordination+multi+agent`, Accessed: 04/25/2010

domain they should be used in to be applicable. This can be specified by requirements towards the coordination mechanisms. The requirement driven selection of coordination mechanisms represents an effective way for mechanism selection and is a key issue introduced in this thesis. It enables making the motivation for design decisions of the design and implementation of a coordination mechanism explicit.

To support the implementation of coordination mechanisms a framework (the CoPS framework) has been developed and is presented in this thesis as well. CoPS is the acronym for *Coordination of Planning Systems*. This framework supports the implementation of coordination mechanisms based on direct communication between the entities. To apply the framework to a given scenario the framework has to be localized on two layers. On the first layer of the network activities have to be coordinated. This comprises the specification of the coordination mechanisms, here encoded in form of communication protocols. If an entity wants to join a network, it is a priori informed about the protocols used for coordination by the network members. Based on this information it can specify its negotiation strategy. In this strategy the planning entity defines which information about its abilities it is willing to publish and which concessions it is willing to make and to whom in the network. The second layer is the localization level. The network wide communication protocol have to be adapted to implement the local negotiation strategy. By these two extension points the coordination framework can be used for various coordination protocols. The necessary aspects of the implementation are structured in the CoPS process which is presented in this thesis, too. Based on the evalution of the applicable, implemented coordination mechanisms a final decision for one particular coordination mechanism can be made on a quantitative base.

1.2. Motivating example: Container terminal management

In this introduction we want to give an example for a complex coordination problem of interdependent planning systems. The problem presented here is the container terminal management (CTM). The task of the CTM is to plan required operations that have to be performed to serve container ships within a port. Containers have to be unloaded, stored and other stored containers have to be loaded on the ship. In the literature, e.g., Zhang et al. [ZLW$^+$03] the problem is divided into the sub-fields shown in Figure 1.1. For the management of the container terminal itself, the field of stowage planning of vessels is not part of the problem scope. This information is typically assumed to be given. This is reasonable as the stowage planning is within the responsibility of the shipping company and not the company running the container terminal. The implementation of the generated

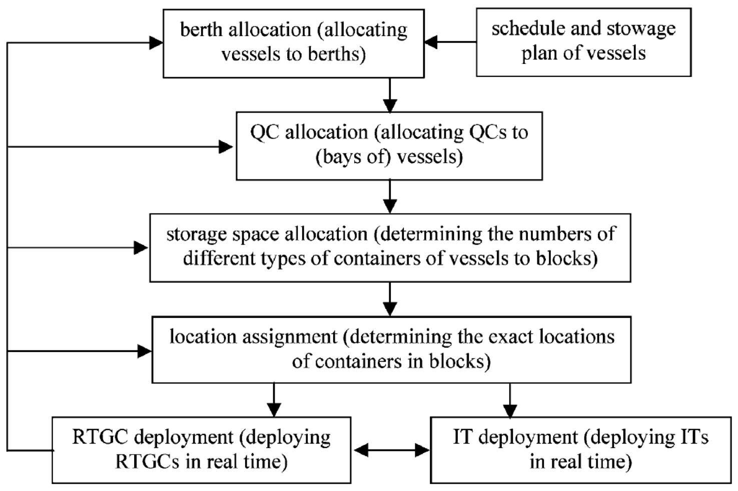

berth allocation (allocating vessels to berths)
schedule and stowage plan of vessels
QC allocation (allocating QCs to (bays of) vessels)
storage space allocation (determining the numbers of different types of containers of vessels to blocks)
location assignment (determining the exact locations of containers in blocks)
RTGC deployment (deploying RTGCs in real time)
IT deployment (deploying ITs in real time)

Figure 1.1.: Decision fields in the container terminal; [ZLW+03, p. 886]

plans are not regarded as fields of the container terminal management problem [Mei08, p. 80]. Which results in the fact that no reactive planning capabilities are integrated. The container terminal operations are often divided into the following four sub-processes:

1. Ship arrival: When a ship arrives the CTM has to locate a berth position. The berth allocation problem (BAP) has to answer the questions when and where the arriving ship should be placed at the berth.

2. Loading and Unloading: Quay cranes and transport equipment have to be allocated to ships for loading and unloading of containers. This allocation problem is also called quay cranes scheduling problem (CSP).

3. Storage location problem (SLP): Containers are stored on the yard. They stay at the yard until they leave the terminal (by ship or truck). So for all unloaded containers storage space has to be allocated at the yard. The storage space allocation comprises decisions concerning stacking density, yard stack configuration, and container allocation.

4. Transportation planning problem (TPP): An effective transport of containers from and to the cranes is an important aspect as the cranes are often bottleneck resources. This transportation planning comprises planning the trucks transporting the containers, thus routing and truck assignment to cranes are exemplified problems that have to be solved. This problem is also referred to as vehicle dispatching problem (VDP).

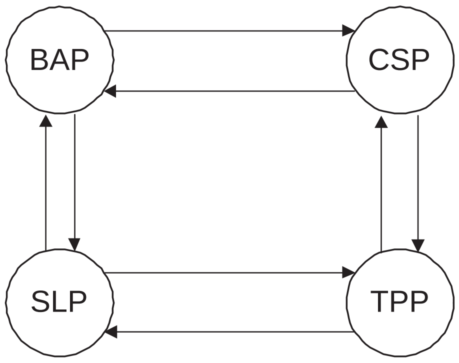


Figure 1.2.: Dependencies of the decision fields in the container terminal, [MS07]

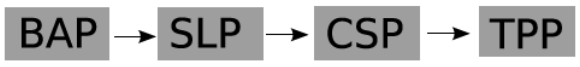


Figure 1.3.: Linear planning sequence of the CTMP

More detailed descriptions of the CTMP can be found, for example in Henesey et al. [HDP06] or Meier [Mei08].

Obviously, the planning tasks of the different sub-processes are interdependent. Their interdependencies have been described in Meier and Schumann [MS07] and are shown in Figure 1.2.

As pointed out by Meier [Mei08, p. 80, p. 220] and in accordance to the model presented by Zhang et al. [ZLW$^+$03] (see Figure 1.1) the planning process is typically performed in a linear manner. Therefore, required inputs that are not given by previous planning steps are estimated, like the service time of the ships for the berth allocation planning. The sequence in which the planning steps are performed is shown in Figure 1.3.

The estimated vales, e.g., for service times of the ships do not have to be correct, as the result of later planning steps may allow to state these values more precisely. For instance, the service time of a vessel depends heavily on the crane scheduling, which is done after the berth allocation. Consequently, inefficient or infeasible plans are the result of this simplified sequential planning process, omitting existing dependencies.

To generate a feasible schedule, the different planning steps have to be performed more coordinated. This is a particular hard task, as it is not possible simply to allow feedback information. For instance, after the crane scheduling

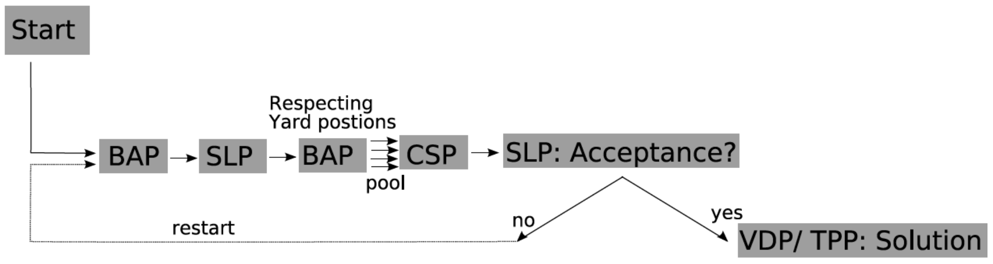


Figure 1.4.: Planning process of the CTMP; [Mei08, p. 222]

the service times are known for each vessel, an information required in a previous planning step. So it is not reasonable to simply restart the berth allocation planning system with these computed service times as inputs. The resulting berth allocation plan would result in a completely different crane schedule and thus different service times. Results from Meier and Schumann [MS07] supports this. In our experiments we performed iteratively the planning steps for BAP and CSP, and found no indications for convergence concerning any of our metrics.

To allow for a more consistent planning, Meier has proposed a different planning process in his dissertation [Mei08]. He defines a planning process in a way that some planning steps are done more often. Moreover, the BAP planner can offer a set of solutions to the subsequent planning steps. He follows the approach suggested in [MS07] to represent each planning system by an agent, which then coordinates their local plan. The process defined by Meier [Mei08, p. 222] is shown in Figure 1.4. After an initial run of the BAP and SLP planning, determining an initial berth allocation and an initial allocation of containers on the yard are computed. Based on the computed storage allocation the berth allocation is adjusted and a number of BAP solutions are generated that are all acceptable to the BAP planner. In the following crane scheduling step one of these berth allocation plans is selected and an appropriate crane schedule is computed. Subsequently, it is checked if the resulting changes of the storage allocation plan are acceptable. If so, the transportation planning problem is solved in the final planning step. Otherwise the planning process is restarted. For more details of this planning process see Meier [Mei08, Chap. 6.1].

The coordination problem within the CTMP is a particular difficult one, in fact its computational hard. That is and will remain an open research issue in the near future. Its complexity comes from two aspects: first, the sub-problems are highly interdependent, and second, each sub-problem is a complex optimization problem in itself. We use this example in this study, as it allows us to illustrate aspects of the coordination of interdependent planning entities.

1.3. Context, scope, and findings of this study

This study is focussed on the coordination of existing planning systems. Therefore, we take a research perspective primarily from the fields of Artificial Intelligence (AI) and software engineering. Findings of other research areas are mentioned as well, but are not primarily focussed here.

Commonly, the terms planning and scheduling are used in a distinct way in AI. Often planning is associated with problems like choosing and sequencing actions, like STRIPS planning (see Section 2.2.1). Scheduling is seen as planning with time. Actions have a duration and typically requires resources either consumable or non-consumable. A well-known instance is the job shop scheduling problem. As we deal here with different planning and scheduling problems the term planning is used as a more generic term within this thesis.

The global and local planning problems discussed here can be formulated as optimization problems, too. The terms planning and optimization often are used interchangeable. In the terminology of this study the term optimization is not used because it is often used in the mathematical meaning of optimization, which does not apply here exactly. We do so because the optimality of a solutions is not emphasized as the only criteria. Moreover, the solution quality is an important aspect among others, like computation time or computational resources, for instance. We assume that local planning systems are capable of finding good, even though not optimal solutions, which is true for heuristics which are mostly applied in real world planning systems.

It is assumed in this thesis that a global planning problem exists that is inherently distributed. The distribution among planning entities is assumed to be pre-existing and not in the focus of this study.

It is not intended here to follow an approach that integrates the sub-problems in favor of an integrated solution method. In the focus of our research is the coordination of planning systems that are responsible for the operative planning of activities within organizational entities. We do not explicitly address planning systems for tactical or strategic decision support. Even though the ECo-CoPS approach can be used for the selection of coordination mechanisms for these problems as well, we do not address them here specifically. Planning systems are embedded in an organizational entity that is responsible for the planning task and the resulting plan, and uses the planning system to generate the plan. These entities are referred to as planning entities or planning authorities within this study. In fact, if we are going to address the coordination of plans we have to coordinate the activities of the planning entities. Thus, these planning entities are first class entities in our system model and are represented by agents. We do not argue here in favor of a complete agentification of all concepts but only for the first class entities

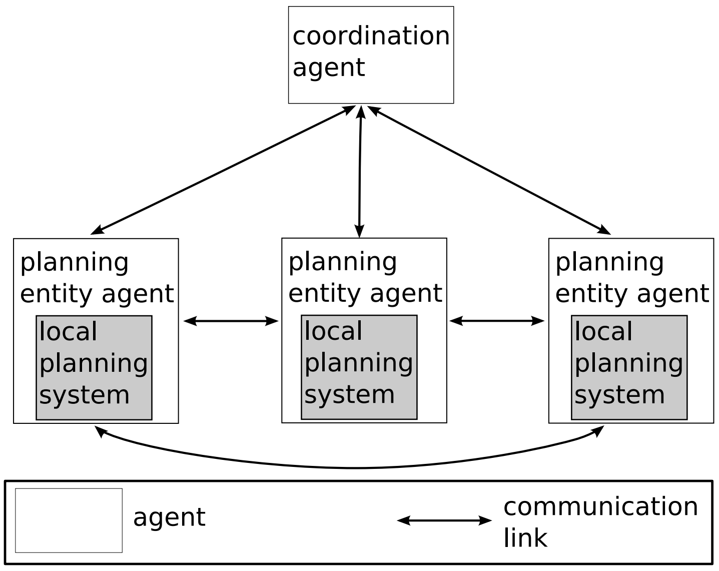


Figure 1.5.: Network layout draft

in our model. If a number of planning entities have to coordinate their activities, i.e., their plans, then they form a network. Typically, such a network has been established beforehand on management level. We advocate that for each network a coordination agent can offer centralized infrastructure services like registration and bookkeeping. This avoids a number of broadcasts that would be necessary otherwise. If this *coordination agent* is a trusted third party the confidentiality level within the network can be improved, too. Moreover, we assume that agents can communicate via direct message passing. The overall layout of a network as pointed out here is shown in Figure 1.5.

The agents form an additional layer in the software system that offers additional services. They enable the multiagent system to coordinate the local plans generated by the local planning systems. Thereby, we strive towards the generation of plans that are feasible for a joint execution. Aspects like the quality of a plan or the joint plan are not effected, as there is no guarantee that a certain overall or local quality can be achieved.

After we have outlined the context we discuss the major findings of this study. We have investigated existing work done so far concerning the reuse within the field of agent-oriented software engineering and mechanism selection in software engineering. As a result of our survey we have to state the reuse of existing mechanisms in agent-oriented software engineering has not been addressed intensively so far. For instance, no process exist for the selection of coordination mechanisms for a given problem, also, other reuse aspects have not been addressed, either.

Within this thesis we give a wide overview of existing coordination techniques that have been proposed in DAI and also related fields of research like game theory, economics, and management. Furthermore, existing classification schemes for coordination mechanisms have been surveyed. Classification is either done by methodological [SSWG02], perspective [Sch01], or on the basis of general models criteria [BCGZ01].

The ECo process, presented in this thesis, provides a systematic approach for the selection of an existing coordination mechanism for a given context. Thereby, we identify the specification of coordination requirements as a key concept. If coordination mechanisms can be classified according to requirements they either fulfill or fail, the selection process can be facilitated. The ECo process structures the process of the selection of a coordination mechanism, which is currently not supported sufficiently by engineering methodologies for distributed systems, in particular agent-oriented software engineering.

Results of a generalized survey concerning coordination techniques and, typically, characteristics of coordination problems are pointed out. They can be used for the identification of suitable coordination mechanisms and, therefore, can guide the concrete selection of a mechanism. These results have been used in the case studies presented in this study and allow us to reduce the set of possible candidate mechanisms considerably fast.

For the implementation of coordination approaches the CoPS process is suggested here. It addresses the implementation of coordination mechanisms from an engineering perspective. The CoPS process enables an integrated systematical approach that guides the implementation of the selected coordination mechanisms. Additionally, the CoPS process is supported by the CoPS framework which can be used to ease the implementation process, as it offers a platform for the implementation of agents that can coordinate their local plans.

Within the CoPS process, techniques for the design and engineering of conversations in the context of agent-oriented software engineering, are discussed comprehensively. The design of conversations is an issue that has a much more general scope than the specification of message encodings as discussed by the FIPA[5]. The design and implementation of conversations between agents that is discussed in the CoPS process is not limited to the coordination of planning systems. It is suitable for the design and implementation for all agent-based systems in general.

The ECo-CoPS approach that is suggested in this study is applied exemplary using two case studies from the management/logistic domain. In the first case

[5]FIPA is the acronym for the IEEE standard organization for the "Foundation of Intelligent Physical Agents" `http://www.fipa.org`, Accessed: 05/17/2010.

study the planning systems for production, packaging and transportation of goods have to be coordinated. In the second case study the coordination of planning systems of entities forming a production network is addressed.

1.4. Organization of this study

This thesis is structured as follows. In the next chapter (Chapter 2) the fundamentals of this study are introduced. This comprises the introduction of the case studies (Section 2.1) that are used within this study, exemplifying certain aspects of the coordination problem, but also are used for the validation of the ECo-CoPS approach. The fundamentals of planning problems are introduced in Section 2.2. In particular, we are addressing the modeling of planning problems (Section 2.2.1), examples of different planning problems that could be effected by coordination efforts (Section 2.2.2), the modeling of dynamics and its effects to the problem solving procedure (Section 2.2.3) and finally we discuss dependencies between planning systems. Note that techniques for solving planning and optimization problems are not discussed in this thesis. This study does not contribute to the solution of planning problems, at all.

A second foundation of this study is the field of DAI although termed multiagent system (MAS). Basis concepts are introduced in Section 2.3. The aspects of communication and interactions between agents is presented in Section 2.3.4.

After the foundation of this study has been laid out the state of the art is discussed (Chapter 3). The state of the art presented in this study covers the field of coordination (Section 3.1), and the field of agent-oriented software engineering (Section 3.2).

The research concerning coordination is a multidisciplinary one. Coordination is addressed in different fields of research, including computer science and organizational science, for instance. We review the state of the art in the field of DAI (Section 3.1.2) but also address the research concerning coordination in the related research fields of business administration (Section 3.1.1) and game theory including mechanism design (Section 3.1.3).

The last part of the state of the art addresses the field of agent-oriented software engineering (AOSE). We survey the field of agent-oriented software engineering in Section 3.2.1. Then particular aspect of reuse strategies within the field are discussed (Section 3.2.2). A result of our survey is, that the reuse within the field of AOSE is primarily limited to the usage of platforms for agent development. Almost no concepts and methods for the reuse of mechanisms exist that can be incorporated into agents. Consequently, neither no approaches exist for the efficient identification and selection of coordination mechanisms for planning agents or agents that represent autonomous planning entities.

As a consequence of our findings surveying existing research, we propose the ECo-CoPS approach in the following chapter (Chapter 4).

A first step to an efficient identification of coordination mechanism is their classification. In the first section of the chapter (Section 4.1) we point out six characteristics of coordination scenarios that have great impact on the applicability of coordination mechanisms. These characteristics depend on the local planning systems, their dependencies among each other, and on context information, as well. Based on this characterization, the coordination mechanisms are classified in accordance to their abilities to be applicable concerning given attributes of the characterization. This allows a first and fast identification of coordination mechanisms candidates for a given situation that can be investigated in more detail.

Then we introduce the ECo process that is presented in detail in Section 4.2. Each step of the ECo process is detailed and appropriate examples are given. For the implementation process of the coordination mechanisms, which is a step of the ECo process, the CoPS process is presented. The goal of the CoPS process is to guide the implementation decisions that have to be made during the implementation of the selected coordination mechanisms. To further ease the work of the developer, the CoPS framework is proposed that allows to ease the implementation process by providing basic features like easier protocol specification and runtime adaptation of conversation protocols. The CoPS process and framework are detailed in Section 4.3.

In the following chapters the ECo-CoPS approach is validated. This is done by presenting the applicability and feasibility of the ECo-CoPS approach. We apply the ECo-CoPS appraoch in two case studies presented previously in Section 2.1. In Chapter 5 the coordination of three different planning systems for scheduling, packaging, and transportation is demonstrated, while in Chapter 6 the coordination of different planning entities within a production network is demonstrated.

Finally, in Chapter 7 a conclusion is drawn. The findings and results of this study are summarized and future research is outlined.

In the appendices additional detailed information concerning the representation of conversation protocols (Appendix A) and the case studies used in the validation (appendix B and C) are given.

2. Foundations and principles

In this chapter the foundation used in this study are presented. At first, we introduce the case studies that are used to exemplify different aspects of coordination between autonomous planning entities. For that reason, we present two case studies from the logistic domain. Then the two sub-disciplines from the field of AI that this word is based on are presented. These are planning and scheduling and DAI.

2.1. The case studies

In this section the case studies used in this study are introduced. They are taken from the domain of production and logistics, not because the problem of coordinating autonomous entities is limited to those domains, but it allows us to outline and demonstrate the problems clearly. The first case study is motivated by a situation in a mid-sized company, covering production and distribution of goods. The second case study reflects the situation in a production network/supply chain where different companies have to cooperate to achieve a common goal and thereby coordinate their activities to perform them cost-efficient and stay competitive. These case studies covers different kinds of coordination problems that can exist between planning systems. These examples differ in their complexity and in types of dependencies that exist between the planning entities.

2.1.1. Production scheduling and distribution of goods

Suppose a factory that applies a build to order strategy. The production schedule is generated based on the given set of orders. An order specifies the quantity of a certain product that has to be delivered to the customer within a given time frame. If this frame is missed, e.g., by late delivery, a penalty has to be paid per each time unit the deadline is missed. It is allowed to fulfill the order by partial deliveries. An order is fulfilled if all ordered products have arrived at the customer site.

As the resources for the production are expensive they have to be used efficiently to gain a high utilization of the resources. To achieve this goal a scheduling system is used that optimizes the utilization of the existing equipment. To detail

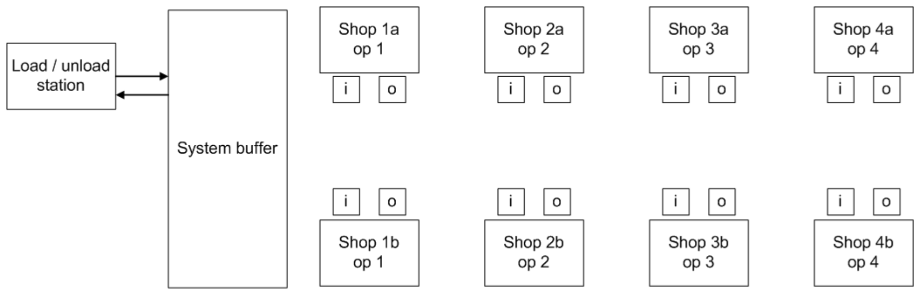


Figure 2.1.: Shop layout, according to [CBM+99]

the production environment, assume the shop layout shown in Figure 2.1. Each shop has an input and an output buffer. It offers exactly one operation. For simplicity reasons transportation within the shop floor is not modeled explicitly. It is assumed that enough transport capacity is always available and transportation time is zero.

The job characteristics were taken from Brennan and O [BO00]. In Table 2.1 the duration and shop sequences are summarized. Five different job types exist, which

Step/job type	1	2	3	4
J1	6/1	8/2	13/3	5/4
J2	4/1	3/2	8/3	3/4
J3	3/4	6/2	15/1	4/3
J4	5/2	6/1	13/3	4/4
J5	5/1	3/2	8/4	4/3

Table 2.1.: Processing plan for different jobs, encoded as time/operation, according to [BO00]

differ in their processing time for each operation and the sequence of operations that have to be performed to fulfill a job.

The goods have to be placed on loading devices before they can be shipped to the customer. The placement of the goods on the loading devices defines a packing problem. To ensure an efficient usage of these loading devices a specialized software solution has been implemented to ensure an efficient usage of loading devices. Each product has a given volume and a loading device has a maximum volume. The resulting problem is a 3-D bin packing problem, i.e., the number of loading devices has to be minimized to reduce the costs induced by those devices. To simplify matters we assume that only one kind of loading device with fixed volume and costs per device exists.

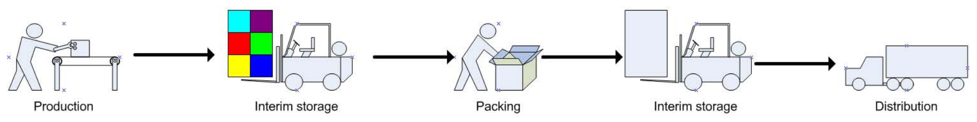


Figure 2.2.: Workflow of a the production and distribution example

After the goods have been packed, they can be shipped to the customers by trucks. Therefore, a homogeneous fleet of trucks is available. Homogeneous indicates here that all trucks have the same capacity for loading devices, the same speed, and a fixed cost rate per distance unit. The resulting transportation problem is to transport the goods within the loaded devices to the corresponding customers. We assume a linear relation between traveling time and distance, i.e., constant speed. The trucks start at the depot in the morning and have to return to the depot after their tour is finished or their maximum hours of working are elapsed. The workflow of this scenario is summarized in Figure 2.2.

This is a typical example for a sequence of steps that have to be performed to serve a customer, which include three different complex planning problems. Each of these optimization problems: scheduling, bin packing, and transportation planning is computational hard and has been subject of research itself. As described, planning systems have been developed and are in place to solve those problems, but these systems are not aware of the context of their application and therefore optimize their plans according to a local objective function. This local optimization will not necessarily lead to a good overall performance.

Similar examples of coordination problems can be found in manufacturing, e.g., in the automobile industry with press, paint, and assembly shops which has been discussed, e.g., by Eppinger et al. [EHNO08]. Each shop has its objective function while the sequence of cars can only change slightly, which makes it a hard task to compute a sequence of cars that have to be produced by the overall system. Sequences leading to a good solution for one shop might cause a miserable performance for another shop.

This rather simplified example shows that the coordination of planning systems can be found in a number of situations and is not limited to multi-national large companies. But as we show in the next section this problem arises there as well.

2.1.2. Supply chain management

The management of the activities of a supply chain has become an entire branch of research concerning the issue of supply chain management [SK02]. The ideal view of a supply chain is depicted in Figure 2.3. There exists one global control

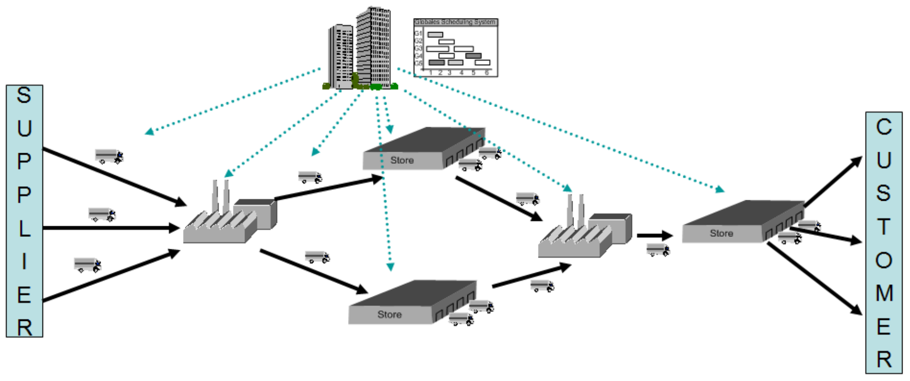


Figure 2.3.: Ideal of an integrated supply chain, on the base of `http://be.wi-ol.de/35677.html`, Accessed: 01/06/2010

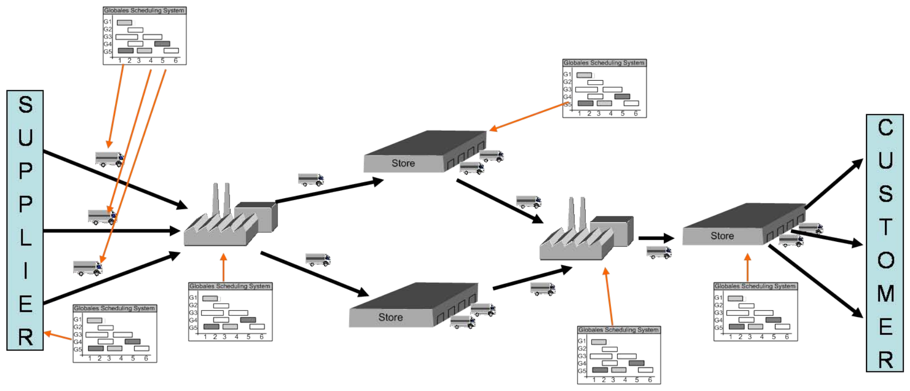


Figure 2.4.: Planning in supply chains, on the base of `http://be.wi-ol.de/35677.html`, Accessed: 01/06/2010

steering all activities within the supply chain. Typically, it is assumed that this control is done by the focal company, i.e., the company with the market power, like an OEM. For such companies additional software, the *advanced planning systems* (APS), have been introduced [SK02] to support these planning activities.

But this ideal view of a supply chain has not been realized, by any means. The companies within a supply chain are autonomous entities in the legal and, sometimes with restriction, economical sense. They can be part of different supply chains with different roles [CG01]. Thus, each entity in a supply chain has its own local planning system optimizing the local profit. Therefore, the situation is more likely to the one shown in Figure 2.4.

Moreover, within a supply chain companies can have overlapping competences

Product	Variant	Operations
1	11	111, 112, 113, 114
2	21	211, 212, 213, 214

Table 2.2.: Product descriptions of the production network

company	offered operations
A	111, 114, 211, 212, 213, 214
B	112, 113, 211, 212, 213, 214

Table 2.3.: Capabilities of production network members

[CG01]. In consequence, the relation between the companies has elements of cooperation and competition, as companies have to cooperate to achieve the goals of the supply chain on the one hand, whereas they have to protect their competence from their competitor on the other hand.

We restrict this case study to only small instance, because each planning problem can become quite complex and time consuming and thus evaluation becomes untractable. In the following, we outline this particular scenario. Our network comprises only two planning entities, who offer different operations to the network. The network is able to generate two different products. For each product only one variant to produce them is defined. A variant is a sequence of operations that can be performed by the supply chain partners. The products offered and their description how to build them is shown in Table 2.2.

An order for the network specifies a certain quantity of one product that has to be produced in a given time window. As pointed out before, each company can offer a different set of operations that are necessary to fulfill incoming orders. In the following Table 2.3 the capabilities of the companies are summarized.

As it becomes clear from Table 2.3, both companies have overlapping competences how to perform operations, even though they may be performed in different ways, i.e., using different resources and vary in the duration of the operation. Moreover, for orders of the product 1 both companies have to cooperate, as each of them cannot perform all necessary operations on their own. Thus, both companies have a relationship with cooperative and competitive elements, as outlined above.

In a production network as sketched here, each operation can be a complex task comprising different planning steps like cutting material, manufacture parts, paint parts, and assemble parts to products. In the perspective of a supply chain, these necessary steps to perform one operation are not detailed any more, as they are part of the execution of one operation that is performed by one supply chain partner. The resulting complexity each supply chain partner has to solve can become quite high, so that each supply chain partner runs a local planning system that details the necessary activities to perform an operation and schedule their execution. Thus, each supply chain partner has a scheduling problem to solve. These local schedules have to be coordinated to ensure that the overall schedule is feasible. It has to contain all operations that are required to fulfill all existing orders in the correct sequence. Moreover, the global plan should try to meet the time windows of orders, if possible.

2.2. Planning, Scheduling and Optimization

In this section an introduction in AI planning and scheduling is provided. While planning and scheduling are fields that are within the scope of AI, the term optimization is more likely used in operations research. Nevertheless, optimization is introduced here as well, as some problems and techniques discussed here are discussed in operations research, too. As we are dealing with the coordination of planning systems in this thesis, the planning systems themselves are not the focus of this study. The techniques for solving those problems are not subject of this study, as well. If appropriate, references will be provided.

2.2.1. Planning, scheduling and optimization: definitions and models

At first, the modeling of planning problems is discussed. Hereby, planning is also used as the more general term that can be used to subsume scheduling and optimization problems. As already mentioned planning and scheduling have different notions in AI. While planning tries to determine the sequence of actions that has to be performed to achieve a certain goal, the task of scheduling is to determine when and how a set of actions should be performed to achieve a goal efficiently. Thus, the activities/operations that have to be scheduled are typically known in advance, but their execution may have to regard resource and time constraints, which is typically the task of scheduling.

Modeling problems

Here we give a short introduction into the modeling of planning problems. General techniques are presented in contrast to domain specific modeling that maybe suitable for a certain domain but cannot be used in a broader field of applications. Examples of more domain specific descriptions are exemplified, e.g., in Section 2.2.2.

Modeling planning problems According to Ghallab et al. [GNT04, p. 19] there exist three different representations for classical planning problems, which are all expressively equivalent. These representations are the:

- set-theoretic representation: A state of the world is a set of propositions. Actions have sets of propositions for their precondition, a set of propositions that it removes, and a set of propositions that are added to the new world state.

- classical representation: States and actions are similar defined as in the set-theoretic representation, but first-order literals and logical connectives instead of propositions are used. This representation is according to Ghallab et al. [GNT04, p. 19] most widespread.

- state-variable representation: A state is represented by a tuple of n attributes. $\{x_1, ...x_n\}$. Actions are formulated as partial functions that map this tuples into some other tuples in the space of the n-tuple space.

In the following, we present the well-known classical representation. The Stanford Research Institute Problem Solver (STRIPS) language, which was one of the first wide spread language for representation of planning problems. Note that it become evident that the expressive power of STRIPS is not sufficient, and therefore other language like the Action Description Language (ADL) have been developed, which extend STRIPS. Both languages can be described out of three components: the states of the environment, the goal(s) that have to be achieved, and the actions that can be executed. A more detailed description of the STRIPS language and a comparison between STRIPS and ADL can be found at Russell and Norvig [RN03, pp. 377–379].

In the STRIPS language states can be represented by logical conditions of the environment in the form of conjunctions of positive literals. Those literals must be grounded and no function expressions are allowed. Note that STRIPS rely on the closed world assumption, i.e., the state of the environment can only be changed by the execution of an action. The goals can be expressed as a conjunction of positive literals. Actions are characterized by their name and a list of parameters. An action has a precondition that must be fulfilled to execute the action. A precondition is a conjunction of function-free positive literals. The effects of an action describe how the state of the world is affected by the action. It is described as a conjunction of function free literals. To improve the readability of the effects of an action the literals are divided into an add list and a delete list. That is literals that are added to the world state belong in the add list, while the other are in the delete list. It is assumed that all literals of a state that are not mentioned in the effects of an action remain unchanged when the action is executed[1].

A language recently developed to describe planning problems is the planning domain definition language (PDDL). PDDL was developed for the international planning competition in 1998 to describe and exchange the planning task [MGH+98]. Therefore, PDDL has been extended and additional expressive power has been added to describe various planning problems. PDDL is currently available in version 3.1 [Hel09].

A very detailed and sound survey regarding the modeling of planning problems is given by Ghallab et al. [GNT04, Chap. 2].

Modeling scheduling problems There exist a number of representations for scheduling problems. They can be characterized either as based on mathematical model or based on set-constraint models. Mathematical model origins from the field of operations research. In contrast set-constraint models have their foundations in AI [Sau02, p. 27].

Nevertheless, both approaches share some communalities in their problem formulation. The communalities are presented here in a way similar to Sauer [Sau93, pp. 21,22]. The task is commonly to schedule n orders on m machines. To generate a schedule a set of operations have to be scheduled. An operation can be described in its simplest form as a tripe (i, j, K), which encodes the operation j of order i that has to be performed on resource/machine k. Each operations has a duration $d(i, j, k)$. Between different operations a precedence relation $<_p$ can exist. If $(i, j, k) <_p (i, j', k')$ holds operation j has to be finished before j' can be started.

Typically a number of assumptions are made to reduce the complexity of scheduling problems. These assumptions are:

- All jobs are known in advance: This assumption can be relaxed for planning in dynamic environments, which is discussed below (Section 2.2.3).
- Operations cannot be interrupted: This is also referred to as non-preemptive scheduling. Preemptive scheduling is not in the scope of this study and therefore not described here.
- Only one resource per operation exist: In the later discussed models these assumption is relaxed and alternative resources are allowed.
- No alternative precedence relations: There exists only one precedence relation per order. In the models presented below variants are introduced, that represent alternative precedence relationships.

[1]This is the so called STRIPS assumption.

values for α	**meaning**
1	one single machine exists
P_m	m parallel identical machines exist
F_m	m machines forming a flow shop problem
O_m	m machines forming an open shop problem
J_m	m machines forming a job shop problem
values for β	**meaning**
r_j	jobs have release dates
$prec$	precedence relations have to be regarded
$premp$	preemptions are allowed
M_j	restrictions for machines exist (e.g., set-up times)
values for γ	**meaning**
L_j	lateness of jobs
T_J	tardiness (i.e., max lateness of a job)

Table 2.4.: Examples of characteristics of scheduling problems

If the precedence relation is equal for all orders the resulting problem is called a flow shop scheduling problem, as all orders flows in the same sequence through the shop floor. If no sequence exists the problem is called an open shop scheduling problem in such a setting. All operations for each order have to be executed, and it is not allowed to do that in parallel but the sequence of the operations does not have to be regarded. If different sequences exist for the orders the problem is called job shop scheduling problem. In such a type of problem each job/order can have its individual precedence relations for its operations.

Mathematical models The most prominent mathematical classification and description means for scheduling problems has been presented by Graham et al. [GLLK79]. A scheduling problem can be described with the triple $(\alpha|\beta|\gamma)$. Hereby α denotes the type of the problem, β indicates the characteristics of the jobs and γ gives the objective function. Examples of values for values for α, β and γ are shown in Table 2.4.

To describe a planning problem one element for the α has to be chosen, whereas a number of elements can be used to describe the β fraction of the description and exactly one objective function has to be specified in the γ part.

A more detailed overview of possible values can be found in [Sau02, pp. 27,28].

Researchers in operations research aim to solve those kinds problems optimally. Unfortunately, most of these problems are NP-hard and an optimal solution cannot be provided even for small problem instances in reasonable time. For instance, the problems 1|release-dates|max-tardiness, $J2$||max-tardiness and $O3$||maxspan are NP-hard [GNT04, p. 359].

Set-constraints-based modeling The set constraint-based modeling rely on the specification of characteristics of a planning problem by describing the relevant entities and their relationships. Different approaches have been proposed that differ in their granularity and expressive power.

According to Keng et al. [KYR88] and Ghallab et al.[GNT04, pp. 351,352] a scheduling problem is characterized by

- a set of resources and their future availability,
- a set of action that have to be scheduled and their required resources,
- a set of constraints that have to be regarded, and
- an objective function.

A more extended characterization has been presented by Sauer, e.g., discussed in [Sau02, Chap. 2.3.2]. Sauer models a planning problem as a 7-tuple: $(\mathbf{R}, \mathbf{P}, \mathbf{A}, \mathbf{HC}, \mathbf{SC}, \mathbf{Z}, \mathbf{E})$. Thereby, the elements are defined as follows:

- $\mathbf{R} = \{r_1, ..., r_n\}$ a set of resources. Each resource is at each point in time available with a given capacity.
- $\mathbf{P} = \{p_1, ..., p_n\}$ a set of products that can be produced. For each product the production knowledge is available. Within this knowledge a set of variants are defined. A variant is defined by a set of operations and a precedence relation. Operations requires resources and have a duration.
- $\mathbf{A} = \{a_1, ..., a_n\}$ a set of orders. An order specifies a product $p \in \mathbf{P}$, the quantity of the demanded product, an earliest start time and a due date.
- $\mathbf{HC} = \{hc_1, ..., hc_n\}$ a set of hard constraints. These constraints have to be guaranteed. Examples of such constraints are, e.g., that only one variant of a product has to be produced, that the precedence relations have to be regarded, or that resource capacity restrictions are satisfied.
- $\mathbf{SC} = \{sc_1, ..., sc_n\}$ a set of soft constraints. These constraints should be regarded. They are used to model economic restrictions. Thus, soft constraints should guide the scheduler.
- $\mathbf{Z} = \{z_1, ..., z_n\}$ a set of objective functions. An objective functions maps a schedule to a real number.

- $\mathbf{E} = \{e_1, ..., e_n\}$ a set of events. Events change the environment which may invalidate the current schedule. Events are discussed in more detail below in Section 2.2.3. Thereby, classes of possible events are defined and not concrete instances.

Modeling of optimization problems The mathematical modeling of scheduling problems has been already mentioned and is not discussed here in detail. In the following a prototype of a linear programming problem formulation is shown.

$$\begin{aligned}
&min/max : c_1x_1 + c_2x_2 + ... + c_nx_n\\
&\text{subject to}\\
&\qquad a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1\\
&\qquad a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n \leq b_2\\
&\qquad \dots\\
&\qquad a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \geq b_i\\
&\qquad x_1, x_2, ..., x_n \geq 0
\end{aligned}$$

If the formulae used in the objective and the subject part have a linear form, the problem is called a linear programming problem. If all variables $< x_1, x_2, ..., x_n >$ have an integer domain the problems becomes significantly harder and is called integer programming problem. If some variables can have real number and other only can have integer values the resulting problem is called mixed-integer programming problem. The advantage to use those modelings is that general purpose solver are available that can find optimal solutions efficiently. On the one hand it can become a difficult task to formulate a given problem in terms of such an optimization problem. On the other hand the solving of such a problem can become a hard task as well. For instance, mixed integer programming is known to be NP-hard. In such situations if it is either to complex to formulate or solve the problem in this more classical formulation for mathematical optimization problems heuristics and in particular metaheuristics are used. Introduction into the field of metaheuristics are provided, e.g., by Talbi [Tal09]. These metaheuristics rely on specific models. If such techniques are used the modeling heavenly depend on the applied solution techniques.

A detailed introduction in optimization modeling and solving techniques is provided, e.g., by Neumann and Morlock [NM04].

Modeling solutions

In this section the modeling of plans is discussed. In classical AI planning a plan is defined as a sequence of actions, that if applied to the initial state will produce a state that will satisfy the conditions specified in the goal description. The notion of a schedule in scheduling is more complex. For each operation that has to be performed an allocation tuple has to be provided. An allocation is defined as (x, i, k_i, m_{nki}). Whereby x is the starting time of operation k_i of order i on the resource m_{nki}.

A schedule S is *feasible* if it satisfies all (hard) constraints. If the used model of the scheduling problem contains soft constraints and the feasible schedule S satisfy those constraints, as well, the schedule is *consistent* . A schedule S is optimal if no other schedule exists with a higher/lower objective value according to the objective function f. So if the goal is to minimize the objective function formally S is optimal iff

$$\forall S' \in \wp(S) : f(S) \leq f(S').$$

2.2.2. Examples for other planning problems

After classical AI planning and scheduling have been presented here, a selection of other planning problems is presented here, as they are used in the subsequent chapters. In particular these are packing and transportation planning.

Cutting and packing problems

Cutting and packing problems constitute two well-known problems, namely the bin packing problem and the knapsack planning problem.

A bin packing problem can be described as the task to place a fixed number of small items into the minimal number of boxes. All items have to be placed. In contrast to the knapsack problem were a finite number of boxes are available and the provided space should be utilized most effectively by trying to maximize the number of small items that can be placed into the boxes.

Dyckhoff [Dyc90] advocates the use of typical characteristics to identify similar groups of cutting and packing problems. These characteristics are:

1. dimensionality (1, 2, 3, n)
2. kind of assignment (B, V)
   - (B) all objects and a selection of items
   - (V) a selection of objects and all items
3. assortment of large objects (O, I, D)
   - (O) one object
   - (I) identical figures

    - (D) different figured

4. assortment of small items (F, M, R, C)
    - (F) few items (of different figures)
    - (M) many items of many different figures
    - (R) many items of relatively few different figures
    - (C) congruent figures

Detailed discussions about this notation are provided by Dyckhoff [Dyc90] and Wäscher et al. [WHS07]. A problem description of a cutting or packing problem is a 4-tuple, describing the problem in respect to the mentioned characteristics.

Packing and cutting packing are manifold and have various different application fields. For instance, the packaging planning of potted plants, as a special form of a 3-D bin packing. This project has been described in Schumann et al. [SPS10].

Transportation planning

The field of transportation planning is currently mainly focused on the field of operations research, even though it is also well studied in computer science. In the field of transportation planning different standardized benchmark problems exist that can be extended for different applications and research issues. Here three base classes and one extension is presented.

The probably most well-known problem class is the traveling salesman problem. The problem can be described as the task to visit n cities. The distances between all cities are known. The task is to find a trip starting from city v_i, comprises all other cities exactly one time, ending the trip in city v_i, and thereby minimizing the traveled distance. A detailed discussion can be found, for instance, in Neumann and Morlock [NM04, Chap. 3.5].

The vehicle routing problem (VRP) is an extension of the traveling salesman problem. Starting from a depot a number of customers $i \in N = \{1, ...n\}$ have to be served with d_i units of a unified good. The task is to minimize the total costs of delivery. Delivery is done by a number of trucks each starting in the depot and must return to the depot, as well. Typically, it is assumed that all trucks are homogenous concerning their capacity and speed. The distances between all customers and the depot are given. A more detailed description is provided, e.g., by Geiger and Wenger [GW07] and Neumann and Morlock [NM04, pp. 468–473]. Geiger and Wenger pointed out clearly that the VRP can be decomposed into a clustering problem, assigning the customers to the trucks, and a routing problem, computing the tours for the trucks.

The pick-up and delivery problem relaxes the assumption that all goods are available at the depot. Instead a number of transportation request have to be fulfilled. Each transportation request specifies a single origin and a single destination. All vehicles start and end their tours at a central depot. A broader discussion of the pickup and delivery problem and related problems is presented, e.g., by Savelsbergh and Sol [SS95].

A specialization of the VRP is the MDVRPTW, that is the multi depot vehicle routing problems with time windows. As indicated by the name there exist not only one but many depots that could serve customers. And each customer has a time window assigned. In this time window the customer has to be served. If hard time windows are assumed a delivery is only possible within the time windows. If soft time windows are assumed a delivery not in the time window will result in a penalty. An example of a solution for a MDVRPTW is shown in Figure 2.5. This figure shows a solution of the problem instance *pr20* by Cordeau et al. [CLM04] computed by a solution developed by Timmermann and Schumann [TS08].

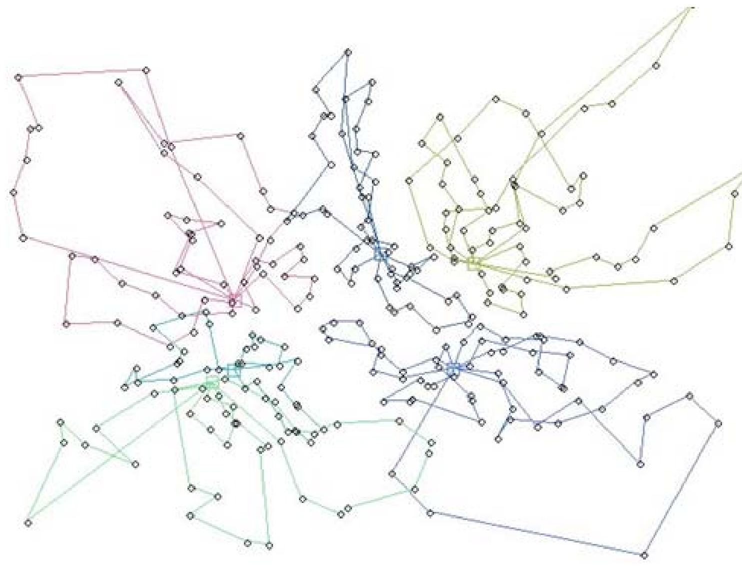


Figure 2.5.: Tours of a MDVRPTW planning instance

2.2.3. Dynamic environments

The environment for planning, scheduling and coordination should allow dynamics. This is a pre-requisite to transfer results into practical applications. In this section the characteristics and consequences of dynamics in planning and scheduling are discussed.

Characterizing dynamics in planning

In nearly all models for reactive planning dynamics is modeled by events that change the environment, and may invalidate the current plan[2]. Those events are often abstractions from data collected in reality. Note that not all events that are observed in reality have to lead to an infeasible plan at all. For example, a resource breakdown of an unscheduled resource does not disturb the plan, but, of course, these events have to be monitored, as it cannot take for granted that this resource is never used[3]. Events can influence the current plan in a positive or negative way. An example of a negative effect on an existing plan is the arrival of a new order. Typically, it is encoded as a hard constraint that all existing orders have to be integrated into the plan. Thus, this event obviously influence negatively the plan, as it becomes invalid. An example of an event that can have positive effects on a plan is the cancelation of an existing order. Different strategies exist how to react on this type of event:

- Ignore the event completely and do not change the plan, at all. Of course there are unnecessary actions within the plan, but commonly the plan will not be decreased[4].
- Delete the assignments planned to fulfill the withdrawn order. Thereby, one cannot decrease the plan. It can eventually improve, if the fulfillment of this order have negative effects on the plan, i.e., the fulfillment of this order was delayed.
- Delete the assignments planned to fulfill the withdrawn order. Then evaluate if other assignments in the existing plan can be changed to improve the plan's quality, by using freed resources to improve the fulfillment of the remaining orders.

Actually, it is an open question to what kinds of events an online planning system has to react. One would expect that online planning reacts when a plan becomes invalid. But a literature survey provided by Schumann and Sauer [SS09], shows that this is not common ground, so far. For example in Larsen [Lar01] only the event of new customers is regarded, which is not the only event that can invalidate a plan. The set of regarded events in transportation planning has been extended by Schumann et al. [STT09] to at least five different events that could invalidate

[2]This is not the only way to handle plan adaption in dynamic environments, of course. In their framework Vieira et al. [VHL03] classify the approaches into event-driven or periodic. In a periodic approach the plan is regularly updated regarding to a rolling time horizon.

[3]This situation only indicates a not efficient investment strategy.

[4]Expect you have some contribution margin elements in your objective function, which is rather infrequent.

the plan. The number of events that can occur is even higher in the job shop scheduling domain, for example Reheja et al. [RRS03] list 17 different events that can occur in their model.

It is arguable to only react on negative events, as the plan's quality is typically decreased by those events, but potentials to improve the plan are not used. Improvement potential often arise as a consequence of positive, i.e., not plan invalidating events.

Reacting to events results typically in change the current plan, which can decrease the stability of a plan. It is only *can* here, because a plan adoption might save future adjustments of the plan, as a consequence of upcoming events in the future. This depends on the characteristics of future events and the event handling strategy. Current discussions on plan stability can be found in [FGLS06, PKM07].

Handling dynamic environments

Planning in dynamic environments also termed planning under uncertainty has been dealt in classical AI planning, scheduling, and optimization approaches as well.

Classical AI planning under uncertainty In most planning approaches presented in the following it is assumed that the planning can start with initial information about the environment that does not change while the planning is performed, but might change during the plan's execution. Some simple approaches have been summarized by Russell and Norvig [RN03, pp. 430 – 449]. These are the following techniques:

- sensorless planning,
- conditional/contingency planning,
- execution, monitoring and replanning
- continuous planning

In sensorless planning no perception about the current plan's execution is provided. Therefore, a plan has to be constructed that satisfy the goals in all possible circumstances, regardless of the changes in the environments that might happen during plan execution.

In conditional planning, also called contingency planning, a plan is computed that provide a plan for different situations. Conditional steps are introduced in the plan. In a conditional step the state of the environment is evaluated. Depending on the received world state the conditional step can branch into a

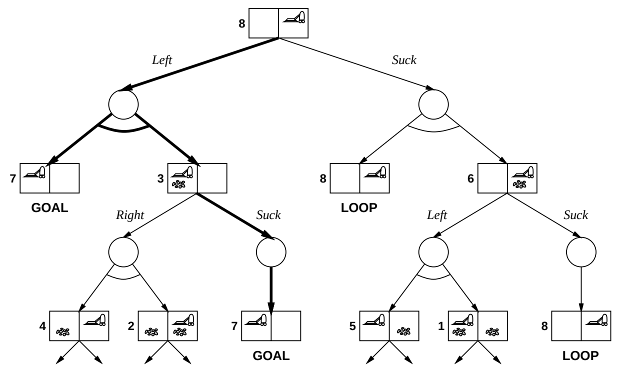


Figure 2.6.: Exemplary conditional plan [RN03, p. 434]

specific sub-plan that is appropriate/specified for this state of the environment. An exemplary conditional schedule from Russell and Norvig is presented in Figure 2.6. Creating a conditional plan is very complex, as all possible states of the environment have to be anticipated and corresponding plans have to be generated. A detailed discussion of conditional planning is provided by Russell and Norvig [RN03, Chap. 12.4]. Ghallab et al. discuss some sophisticated planning techniques for conditional planning using Markov decision processes and model checking for details see [GNT04, Chap. 16, 17].

The idea of the execution, monitoring and execution approach is that a plan is generated, e.g., with one of the techniques already presented. The execution of the plan is monitored. The monitoring can either focus on the next planned action or the entire plan. If only the next action is monitored, it is checked if the next planned action can be executed in the current situation. Otherwise the open plan is monitored. The open plan is the sequence of action that has not been executed so far. This is done by checking all preconditions of the remaining actions that are not generated by previous executable actions. If the monitoring detects a problem, i.e., the next action or the open plan cannot be executed successfully, a replanning is triggered. During the replanning a new plan is generated that tries to make the existing plan executable again or an alternative plan is generated that leads to the desired goals. Note that this notion of the term replanning is not consistent with the use of the term replanning below. An elaborated discussion about this approach can be found in Russell and Norvig [RN03, Chap. 12.5].

The idea of continuous planning is to interleave planning and execution within an infinite loop. The planner does not terminate when a certain goal is satisfied, so it can handle also dynamic addition of goals. The control loop is sketched in Algorithm 1.

Algorithm 1 Control loop of a continuous planner

```
while true do
    monitor environment
    plan
    execute
end while
```

The planner can be, e.g., a partial order planner. During the planning process open preconditions are refined and existing conflicts are resolved. A detailed discussion is proved by Russell and Norvig [RN03, Chap. 12.6].

Scheduling in dynamic environments Before we present the techniques used for handling dynamics in scheduling and optimization, some terminological issues have to be clarified. Especially in the handling of dynamic environments techniques in scheduling and optimization have reached a significant overlap, that justifies to put them together in one section. But as a consequence of their distinct development their terminology is still inconsistent.

Some terminology issues The problem of keeping existing plans in a dynamic environment feasible is addressed in different fields of research and applications. The fields addressed here are operations research and AI. The applications mentioned here are vehicle routing problems and (job-shop) scheduling problems. Different terminologies have been established and terms are used partially interchangeable. Therefore, it is necessary to clarify the terminology. The most general term we use is online planning[5]. Online planning comprises the fields of reactive scheduling and latest commitment planning. The term reactive planning is discussed intensively in Sauer [Sau02, pp. 17–20]. Latest commitment approaches were presented, e.g., by Pollack [Pol96] and Sauter and Parunak [SP99]. Interestingly mechanisms based on latest commitment strategies do not play an important role in current research regarding online planning, even if they are used in practical applications [FHK$^+$07].

Technically there exist two major approaches how reactive planning can be implemented [FGLS06]:

- Plan repair: An existing plan is going to be adapted to a changed situation. Approaches are presented, e.g., in [Smi94, AGO08].

[5]The background for the term online and offline planning are presented, e.g., by Ghallab et al. [GNT04, pp. 14–15].

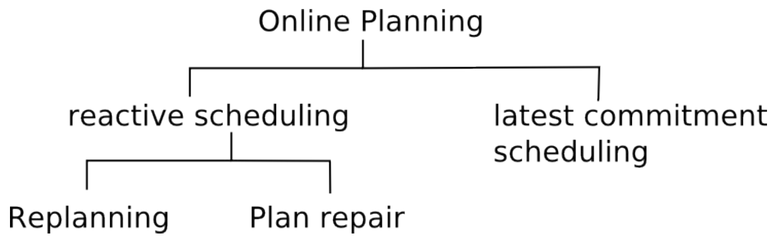


Figure 2.7.: A taxonomy for online planning

- Replanning: If an event occurs the existing plan is discarded and a new plan is computed from scratch [CK05].

Both approaches have their strengths and weaknesses. From the perspective of complexity it has been shown by Nebel and Koehler [NK95] that both approaches face the same problem structure. A comparison of both approaches emphasizing stability can be found in [FGLS06].

Summarizing the terminology discussed above, the taxonomy of terms in the field of online planning is shown in Figure 2.7. It has to be mentioned that this terminology outlined here is not exclusive. There are different naming schemes in use. For example, in Vieira et al. [VHL03] the most general term used is rescheduling. The authors present an interesting framework for rescheduling approaches that can be classified in their framework. Thereby, they distinguish between dynamic scheduling and predictive-reactive scheduling. Dynamic scheduling is characterize situations where no plan exist a priori, like in latest commitment strategies. Furthermore the authors mention that other names for dynamic scheduling are online scheduling or reactive scheduling.

Techniques for handling online planning The goals of reactive planning are according to Dorn [Dor04] threefold:

1. The reaction to an event should be fast. The current plan should become feasible quickly. This is motivated by two reasons. First the plan execution should not be stopped (to long) until a new feasible plan is available. Ideally the execution is going on, while the plan for future activities is adapted. Second in a dynamic environment, like the shop floor or the transportation domain, events are rather frequent. If the system would react to slow the *new* plan is overtaken by reality as it is computed.

2. The existing plan should be widely conserved. This is required as it is intended that steps that have already been performed, should be considered

in the new plan, if possible. So work done so far should not be discarded. Moreover, this enforces plan stability that is important for the coordination of the plan with other planning problems, like procurement, and for the reputation of the planning systems.

3. The third goal is to maintain the plan's quality. A key motivation for using enhanced planning systems is, of course, the ability to compute plans of good quality in short time. Lower evaluated plans lead to inefficient transportation or production and in consequence to operational loss of the company. It goes without saying that this has to be avoided. Therefore, to be practical applicable planning systems have to perform at least acceptable and comparable to human planners in dynamic environments which is measured in plan quality.

As previously outlined we can identify different techniques for reactive scheduling. Therefore, we can identify different technologies for plan repair and replanning, even if there exist technologies that can be applied for both approaches, e.g., metaheuristics.

In the following we are going to discuss techniques that have been applied to plan repair. An important techniques is to use local search techniques. Thereby, plan modification actions are tested to modify the given plan at one position at a time. A typical plan modification is the shifting of dispatched operations in the future (right shift) or pull them backwards in time (left shift). An approach using those techniques is implemented in the OPIS system presented by Smith [Smi94]. Based on local search techniques, metaheuristics can be applied to plan repair as well. A counterargument therefore is the longer reaction time. But metaheuristics strive fast to good solutions, so that the computation can be interrupted at a given time and results can be of good quality. A typical trend of metaheuristics for their plan's quality over time is sketched in Figure 2.8[6]. An approach using tabu-search and genetic algorithms for plan repair has been presented by Dorn et al. [DGSS96]. Other approaches, that have been applied to plan repair are multiagent systems and concepts from the field of swarm intelligence.

Using the characteristics that agents can adopt their behavior and their plans to their environment has motivated the usage of multiagent systems for planning in dynamic environments. Often orders and resources are represented by agents. Agents negotiate about the assignments. Typically, orders want to be processed as quickly as possible while resources want to maximize their utilization. Examples for the usage of agents in dynamic environments in the manufacturing domain can be found in [CGMT00, SL95, WW06b], for instance. Applications in the transportation domain have been presented by [FMPS95, Pol96], for example.

[6]This figure was provided by Tjorben Bogon, also affiliated at the IS-group in Frankfurt.

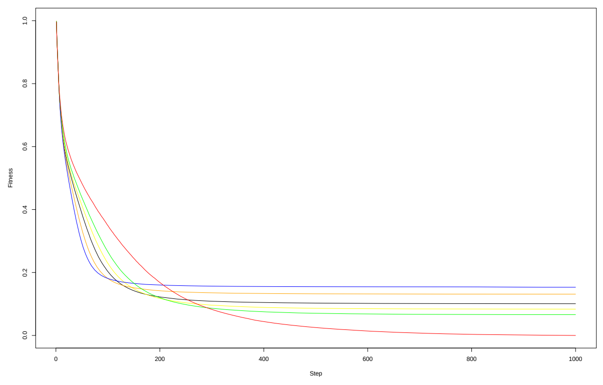


Figure 2.8.: Characteristic plan's quality trend over time, using different parameter sets

Agent-based approaches have also been combined with ant algorithms, e.g., by Heeren [Hee06] or Verstrate and Valckenaers [VV06]. Thereby, the main idea is that if an event changes the planning environment that cannot be handled locally by an agent, this agent starts a plan repair sequence that uses concepts of ant algorithms. The ants collect local information of different authorities to find 'paths' for an effective production.

For the field of replanning different techniques have been investigated. A complete replanning compromise the generation of a new plan for all activities not executed so far. In contrast to partial replanning where only a subset of activities is rescheduled. Other parts of the plan remain fixed and scheduled resources are marked as blocked. After a new schedule has been computed, which can be done faster as the initial computation, because the number of activities to be scheduled have been reduced, the unchanged elements of the plan and the newly computed plan can be joined. The quite common technique to remove orders that are affected by an event, update those data, and then integrate those events as new orders is a special case of partial replanning. According to Pfeiffer et al. [PKM07] complete replanning leads to better plan's quality than partial replanning.
Of course, looking at aspects, like plan stability, complete replanning is problematic, as the plan can change with each event completely. This effect has been described as chaotic planning behavior by Henseler [Hen98, p. 31]. Chaotic planning behavior is characterized by the fact that small changes in the input of a planning system can lead to completely different plans or can result in no changes at all. Chaotic planning behavior can be observed either by predictive planning and reactive planning [Hen98, p. 37]. One can try to soften those effects by adding

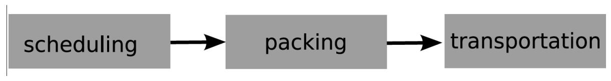


Figure 2.9.: Dependency graph of the SPT scenario

new constraints to the planning problem that fixes parts of the plan in favor of stability. This decreases typically the ability to find feasible plans at all, because it is unknown to what degree similarity can be archived at all.

2.2.4. Dependencies among planning problems

In this part of the study we discuss the modeling of dependencies for analytical purposes. These analyses allow us to gain a fast understanding of the dependencies of planning systems that have to be coordinated. Note that it does not allow us to automatically reason about the dependencies, but it gives an intuitive and fast understanding of the coordination problem at hand, and enables to find appropriate mechanisms for the coordination problem.

According to Meier and Schumann [MS07] dependencies between planning systems can be modeled using dependency graphs. A planning system is, from a very abstract point of view, an algorithm that transforms an input I to an output O concerning a set of constraints C. A dependency exists between two planning systems a, b iff the output of a has an effects on the input or constraints of b. The dependency graph is defined as follows:

$$\begin{aligned} G =& \ (V, E), \\ V =& \ \{\text{A set of nodes, each node representing a planning entity}\}, \text{ and} \\ E =& \ \{\text{A set of dependency edges, an edge connects two nodes } (a, b), \text{ iff} \\ & \ \text{there exist a dependency relation as previously defined}\}. \end{aligned}$$

A simple example of a dependency graph is shown in Figure 2.9. It depicts the dependencies of the production scheduling and distribution of goods example, presented in Section 2.1.1. The dependencies are defined by the material flow that is linear in the system. Note that SPT is the abbreviation for scheduling, packing, and transportation.

Suppose that we have a more intelligent packing planning entity that tries to put together groups of customers that are located in the same region. This strategy depends on information of the transportation planning, i.e., the regions that are typically used to cluster the transportation request to different routes. In such a situation the resulting dependency graph is shown in Figure 2.10.

Another example of a dependencies graph is given in Figure 2.11 for the container terminal management example presented in Section 1.2 as motivating ex-

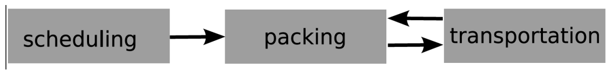


Figure 2.10.: Extended dependency graph of the SPT scenario

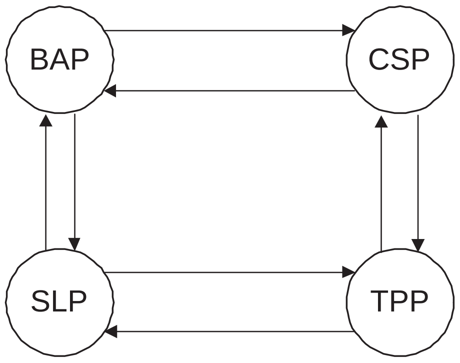


Figure 2.11.: Dependency graph of the container terminal management problem, [MS07]

ample. A detailed discussion of the dependencies of the CTM example can be found, for instance, in Meier and Schumann [MS07] or Meier [Mei08, Chap. 3.3].

The definition of a dependency graph is a rather abstract one, aiming to get a fast overview and awareness of existing dependencies. Its definition might be extended towards an arc-labeled graph in a way that each arc specifies the dependencies. Thus, a label might look like `output/input` or `output/constraint` tuple.

Another technique to encode the dependencies in a machine understandable way is the use of ontologies for representing the planning problems and their plans. Ontology alignment technologies can be used to express the dependencies between corresponding/dependent concepts in the models of the isolated problems. To allow a fast analysis of dependencies either existing ontologies can be used, e.g., for scheduling by Pries [Pri08] or specialized ontologies can be derived from an abstract ontology. In those, more abstract ontologies concepts are provided that could be found in different planning problem. An example for such an abstract ontology has been developed by Koutis [Kou08]. His ontology is an abstraction of ontologies for scheduling, packing and transportation planning problems. The approach using ontologies and ontology alignment is not used in this thesis for the following reasons: Ontologies for planning problems, like scheduling have been proposed in the literature, e.g., by Smith et al. or Pries [SB97, Pri08]. But for more specialized problems no widely accepted ontological representation are available. Moreover severe, the field of ontology alignment and ontology matching is itself a field of research, that is currently evolving. Especially as the alignment technologies do not have to identify the same concepts in this application scenario but have to identify dependencies, which is a more complicated task.

Implications of interdependencies between planning systems

If a dependency exists between two planning systems the need for coordination become obvious. Of particular interest are interdependencies, i.e., cycles in the dependency graph. As already mentioned, in Section 3.1.1 these interdependencies are often not regarded in sequential planning processes that have been suggested by researchers from the field of business administrative research. The iterative plan formation (see Section 3.1.2) is not a solution for handling interdependencies because of the chaotic behavior planning system can exhibit, as experiments from Meier and Schumann [MS07] gives no indication that such an approach leas to converging between plans for the iteratively planning BAP and CSP. This observation is supported by the following theoretical consideration. A naive approach is to estimate the input data of the first planning system and use the results of this system as input of the second one. Then, if the second planning system has computed its plan, derive the required information and use them as input for the first planning system. The underlying idea of this procedure is to compute a fix-point. To ensure termination of this process it is necessary to ensure that the coordinated schedules converge to a stable state (the fix-point) [MS07]. But scheduling problems can be chaotic, as already mentioned. Thus, in the general case it is not possible to show that iteratively coordinated scheduling system convergence towards a stable state. Consequently, cyclic dependencies have to be dealt with. They have to be respected and are not only of academic interest, as shown in the aforementioned examples. Note that not all planning systems are chaotic. Thus, iteratively coordination can work if no such chaotic behavior is present. For instance, Jung and Jeong [JJ05] even provide a proof that their iteration process between two planning steps will terminate.

Complexity of Coordination

Adam et al. [ABTV04] try to estimate the complexity of coordination along three dimensions:

- the number of systems that have to be coordinated,
- the time the systems has to be kept coordinated, and
- the complexity of the systems that has to be coordinated.

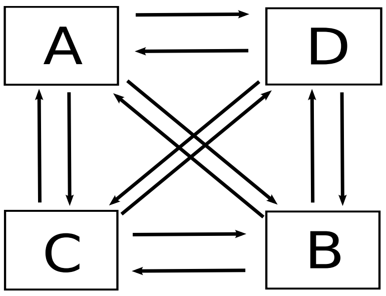


Figure 2.12.: Coordination complexity in a static environment, according to [ABTV04, p. 25]

In the worst case the dependency graph is fully connected. Each planning problem depends on all others. In such a situation each and every entity has to be coordinated with one another. Adam et al. [ABTV04, pp. 25,26] present a graph structure to visualize these coordination dependencies that is very similar to the aforementioned definition of dependency graphs, even though they provide no formal definition for the graph. An example of such a dependency graph is shown in Figure 2.12.

The complexity resulting from dynamics can be modeled by extending the aforementioned graph. For each point in time, a dependency sub-graph is added with time dependencies as additional edges. Consequently, the complexity grows linear in time. A resulting complexity graph in a dynamic environment is shown in Figure 2.13. The number of edged in this dependency graphs indicates the coordination complexity. Of course, the maximal number of edges in a directed graph grows quadratic to the number of nodes and thus the potential complexity of coordination grows quadratical, too. Adam et al. argue that by introducing organizational structures into a system, i.e., arranging the node in an organizational structure, a tree for example [ABTV04, p. 27], the set of edges is bounded and thus coordination complexity. The needed amount of coordination can be reduced as fewer systems have to be coordinated with each other.

Concerning the complexity of the local systems Adam et al. claim that the need for coordination grows with the complexity of the systems that has to be coordinated exponentially[7] [ABTV04, p. 22].

[7]In German: “dass der Koordinationsbedarf mit dem Komplexitätsgrad des zu koordinierenden Systems exponentiell steigt” [ABTV04, p. 22].

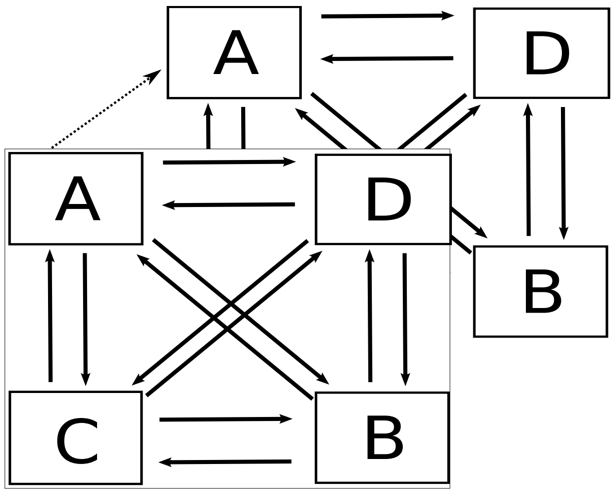


Figure 2.13.: Coordination complexity in a dynamic environment, according to [ABTV04, p. 26]

2.3. Distributed Artificial Intelligence

In this section key concepts of the field of DAI are introduced. Thereby, we start with a single entity, the agent. Agents can be rather simple, like reflex agents, or can be very complex, like intelligent agent. Systems that are composed out of a number of agents are called multiagent systems.

2.3.1. Agents: From reflex to intelligent agents: complexity and architectures

In this section, and for the rest of this study, we use the notion of agency that is primarily motivated by Wooldridge [Woo09]. The author is aware of the fact, that the term agents was subject of discussion itself and a number of other definitions have been proposed, e.g., by Ferber [Fer99]. Nevertheless, the definition by Wooldridge is currently predominant. The definition of an agent by Wooldridge is the following:

> "An *agent* is a computer system that is *situated* in some *environment*, and that is capable of *autonomous action* in this environment in order to meet its delegated objectives." [Woo09, p. 21][8]

This definition emphasize that an agent has orders that are delegated to him, that he is acting on behalf of somebody else. This somebody might be a human or

[8]Adapted version from Wooldridge and Jennings [WJ95].

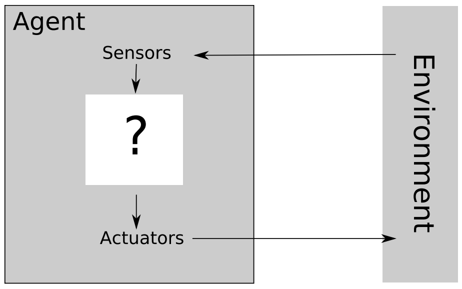


Figure 2.14.: Agents interaction with its environment, according to [RN03, p. 33]

another agent. In the following we refer to this *somebody* as the *principal*[9] of the agent. The aspect of a *situated* agent, also referred to as *embodiment* (discussed for instance by Schumacher [Sch01, pp. 11,12]), is pointed out more precisely in the definition presented by Russell and Norvig [RN03]:

> "An **agent** is anything that can be viewed as perceiving its **environment** through **sensors** and action upon that environment through **actuators**." [RN03, p. 32]

This definition is also referred to by Wooldridge [Woo09, p. 22, Fig.2.1]. Thus, each agent can be described as a system with a perception of its environment and its actions performed on the environment. This is shown in Figure 2.14. Typically, the internal computing, that specify which actions are taken into account is summarized as *reasoning* of the agent. The complexity of the reasoning within the agent can vary drastically. Starting from very simple *reflex agents*, that perform an action according to the given input up to complex agents, also referred to as *intelligent agents*. In this introduction the different levels of complexity are presented, starting with rather simple agent structures and then discuss more complex ones. This introduction base on [RN03, Woo09].

Simple reflex agents

As already mentioned, we start with the simplest structure for agents that are *simple reflex agents*, also referred to as *purely reactive agents*, which react on their current perception, and do not take a history of their earlier perceptions and actions into account. A schematic diagram of a reflex agent is sketched in Figure 2.15. This form of agent has some similarity with simple feedback loops, described in control theory [AM09, Chap. 2]. The delegated objectives that these kinds of

[9]Even if this term is borrowed from the *principal agent theory* it is not intended to restrict the term agent in this study by any means.

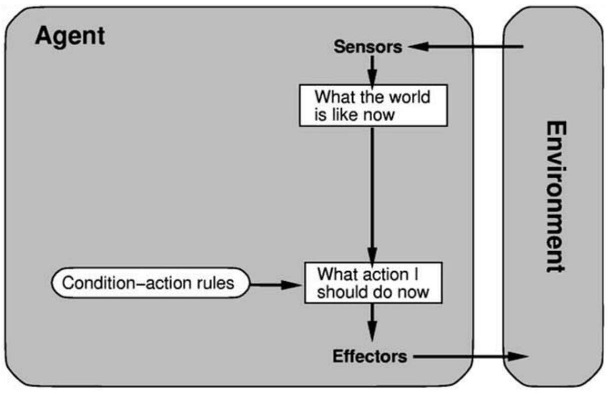


Figure 2.15.: Schematic diagram of a simple reflex agent,[RN03, p. 47]

agents can achieve might be the execution of simple repetitive tasks that can be described in the form of condition-action rules[10]. This kind of agent is simple but from an observers point of view, can only act correctly if the environment is fully observable for the agent [RN03, p. 47]. A more detailed discussion about reflex agents can be found at [RN03, pp. 46–48] or [Woo09, p. 36].

Model-based reflex agents

The next, more complex, form of agents are *model-based reflex agents*. This type of agent has an *internal state* that is updated by the current perception. This state can results from the complete perception sequence of the agent. The next action that has to be executed is chosen according to a set of rules, based on the current (updated) internal state of the agent. Thus, the agent works like a Moore machine (see Hopcroft, Ullman for more details [HU79, p. 42]). The structure of a model-based reflex agent is shown in Figure 2.16. A more detailed discussion can be found at [RN03, pp. 48,49] or [Woo09, p. 37].

Goal-based agent

Based on an internal state that enables to reason about the entire perception sequence the action selection can be improved. This can be done by allowing the agent to reason about its next actions. In the models presented so far, the action selection was done at design-time and encoded in condition-action rules. A more elaborated way is to specify the objective the agent has to satisfy and not only how an agent should behave in a specified situation. This can be stated as *goals* the agent has to achieve. Thereby, the agent is autonomous in deciding how to

[10]Typically, encoded in the form of if-then rules.

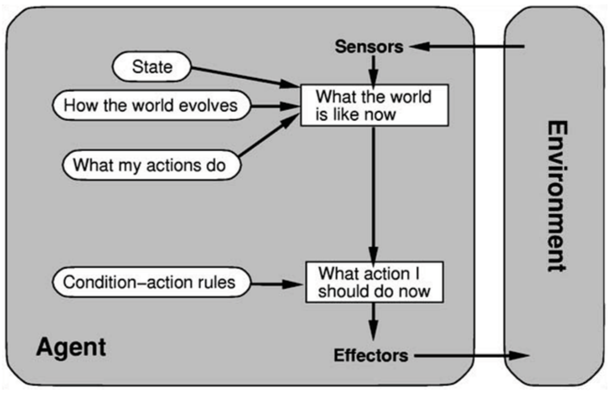


Figure 2.16.: Schematic diagram of a model-based reflex agent [RN03, p. 49]

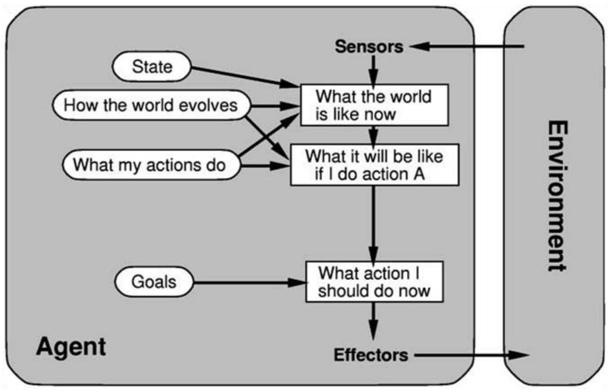

Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Agent
Effectors
Environment

Figure 2.17.: Schematic diagram of a goal-based agent [RN03, p. 50]

achieve these goals. The resulting structure of such an agent is shown in Figure 2.17. The necessary reasoning that an agent has to perform increase the required computational effort drastically in comparison to the types of agents presented so far. It is not only a check of some conditionals to find the rules that apply to the current local state anymore. It can become a complex search or planning problems, which is addressed in AI research itself. This extra effort enables the agent to act much more flexible in contrast to other approaches. It is argued, that this flexibility of goal-based agents and more complex forms of agents offer are a key characteristic for agency in computer science (see for instance Kirn [Kir06]).

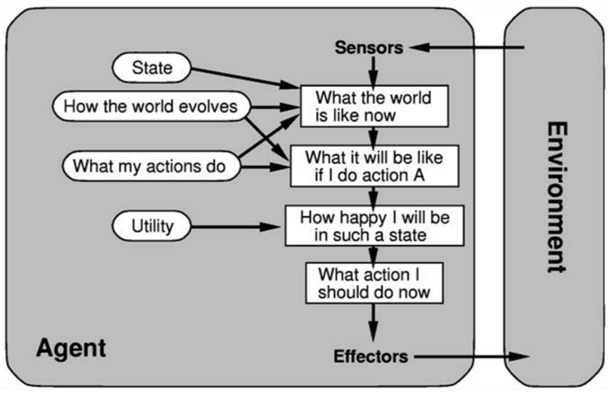

Sensors
State
How the world evolves
What the world is like now
What my actions do
What it will be like if I do action A
Utility
How happy I will be in such a state
What action I should do now
Agent
Effectors
Environment

Figure 2.18.: Schematic diagram of a utility-based agent [RN03, p. 52]

Utility-based agent

By specifying the goal of an agent it remains to the agent how efficient it tries to achieve the goal. Thus, efficiency is out of the scope of goal-based agents. Goals can be either satisfied or not. With a utility function, mapping a state into a real number, the outcome is not binary anymore. Therefore, the next more complex form of an agent is a *utility-based agent*. The utility function enables the agent to measure its performance. So an agent can evaluate its possible actions according to a utility function while trying to maximize its earned utility. The structure of such an agent is shown in Figure 2.18. Utility functions are seen as an efficient way to specify the objectives that are delegated to the agent [Woo09, p. 38].
Utility-based agent are discussed in game theoretic research as well, e.g., as agents that can act in worth oriented domains [RZ98b, Chap. 7]. According to Russell and Norvig the existence of an explicit utility function is a prerequisite for an agent to act rational [RN03, p. 51]. An agent is classified to be a *rational agent* if it compliant with the characterization presented by Russell and Norvig as follows:

> “For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.” [RN03, p. 36]

Other terms used in DAI research for characterizing agents into different taxonomies are according to Wooldridge [Woo09] and Timm [Tim04, p. 17]:

- behavioral agent: an agent that decides what do without reasoning
- deliberative agents: an agent that decides what to do via logical/symbolic reasoning

- cognitive agents: agents with enhanced adaption and learning abilities

Intelligent agent

In this section we introduce the term *intelligent agent*. Here the definition of Wooldridge [Woo09, pp. 26,27][11] is presented. An intelligent agent is an agent that has the following abilities:

- Reactivity: Intelligent agents are able to react on changes in their environment in a timely fashion in a way to satisfy their delegated objectives.
- Proactiveness: Intelligent agent act without an external event to achieve their goals, typically the delegated objectives/goals.
- Social ability: Intelligent agents can interact with other agents to achieve their goals.

Of course, this definition is not exclusive. There exist other definitions for intelligent agents that emphasize other aspects, e.g., the notion of intelligent agents by Nwana [Nwa96], who claims that intelligent agents should incorporate learning abilities. As pointed out before, we focus here on the definition presented by Wooldridge.

2.3.2. Agent Architecture

Depending on the type of agent that has to be build different architectures are suitable. In those architectures the components of the agent's implementation and their interactions are defined. An introduction to software architecture can be found in Taylor et al. [TMD09] or Sommerville [Som06, Chap. 11] for instance.

The selected architecture must be suitable for the intended type of agent. For instance, a utility-based agent cannot be realized using a reactive/reflex agent's architecture. According to Wooldridge [Woo99] four different agent architectures can be classified into:

- Logic-based agents, decision making is done based on logic deduction.
- Reactive agents, decision making is done by a mapping from a situation to an action.
- BDI agents, decision making is done by manipulation of data structures representing beliefs, desires and intentions.

[11]Originally this definition comes from Wooldridge and Jennings [WJ95].

- Layered architectures, decision making is done in a number of software layers. Each layer realizes some reasoning based on parts of, possibly abstracted, environment models.

The idea of purely logic-based agent origins in the traditional symbolic logic oriented AI. Their description is according to Wooldridge [Woo99, pp. 42–48] as follows. The agent decision making is seen as a deduction process. An agent has a theory for specification and it can reason about and execute action based on this reasoning. The main idea of their reasoning mechanism is the cyclic behavior of perception, reasoning (here deduction) and action, as already described in Section 2.3.1. Agents have a database with their current believe set, a set of logic formulae. The perception updates the agent's database, i.e., facts that have become false are removed and others may be added. For the action selection the agents tries to deduce a predicate of the form $Do(a)$ or at least tries to falsify a predicate like $\neg Do(a)$ from his current database and given theory.

Some disadvantages of purely logic-based arise form the fact, that their inference can become computational complex, as automated theorem proving is, in general. Thus if the agent has to regard time constraints, logic-based architecture can become a problem. Especially as it is necessary that the environment stay stable during the inference process within the agent, which restricts the field of applications. Nevertheless, for more theoretical investigations this model is interesting because of its pure logical structure and its resulting clear semantics.

Next the architecture of a reactive agent is discussed. These architectures are not based on symbolic logic, and historically were designed as a consequence of the intractability problems logic-based architectures. These type of architectures was heavily influenced by Brooks[12] [Bro86]. The idea this type of architecture is that an agent can have one or more behavior modules. A behavior module implements a state machine. All these modules have access to the input. Typically, in each state of a behavior module a mapping function exists that maps the input to an activity. In case that more than one behavior module is activated by an input, i.e., there exists a mapping entry within this module for the given input; an action selection process has to be provided. Therefore, the behavior modules are prioritized. So if more than one module is activated the output, i.e., action, from the module with the highest priority is chosen. A more elaborated presentation of this type architectures is provided, e.g., by Wooldridge [Woo99, pp. 48–54].

The main idea of BDI agents is that the knowledge-base of an agent is composed out of three elements, its beliefs, its desires and its intentions. This approach is motivated by psychology model by Bratman [Bra87] and has been formalized for

[12]see Wooldridge [Woo99, p. 48] for historical insights.

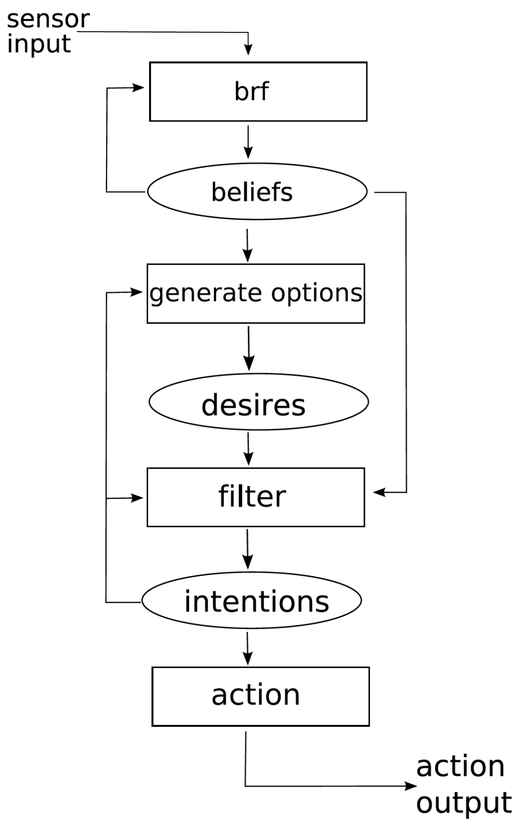


Figure 2.19.: BDI architecture; [Woo99, p. 58]

usage in DAI by Rao and Georgeff [RG95]. Typically, the representation of beliefs, desires and intentions is done using logical formulae, for instance, in first-order logic. The BDI agent architecture is therefore a more practical way to formulate logic-based inferencing agents. The beliefs of an agent can be interpreted as the representation of the current world state the agent believes to be true. While the goals and plans of an agent are stored in the set of desires and intentions respectively. An abstract BDI architecture is shown in Figure 2.19. The function *brf* is the belief revision function, it updates the believe set, according to the current perception of the agent. The function *generate options* computes the current desires based on the current belief set and intentions of the agent. This function of the agent calculates the plans that should be applied to achieve its intentions. Within the *filter* function the agent updates its intentions, for instance, if it is not possible to realize an intention, i.e., there exist no valid plan to do so, it might be reasonable to give up this intention. Based on the updated intentions an executable intention is chosen to be perform a step towards the fulfillment of the selected intention. This is done by the *action*-function. A detailed description this architecture can be found in [Woo99, pp. 58–60]. For the paradigm of BDI agents a number of architectures have been presented, a specific overview about those architectures is provided by Haddadi and Sundermeyer [HS96].

A class of agents, called hybrid agents have layered architectures [Woo09, Chap. 5.2]. There exist different strategies how the components of the agents can be arranged in either horizontal or vertical structures. In a horizontal layering the input is processed by a number of components and then this information is aggregated to determine the output/action of the agent. The information and control flow of a horizontal architecture is shown in Figure 2.20. In contrast to the horizontal

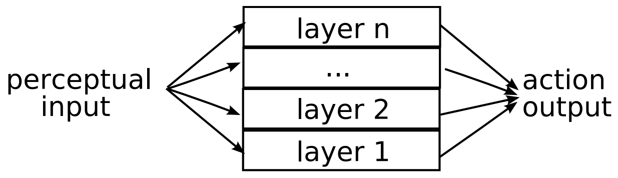


Figure 2.20.: Information and control flow of a horizontal layered architecture; [MPT95, p. 263]

arrangement of components are vertical layered architectures. Vertical layered architectures can be subdivided into one pass control or two pass control architectures. In one pass layered architectures the input is processed in each stage and then delegated to the next one. This architecture is therefore similar to the pipes and filters architecture pattern. See Taylor et al. [TMD09, [p. 110] or Buschmann et al. [BMR$^+$02, pp. 53–70] for a detailed presentation of this pattern. The information and control flow of a vertical one pass architecture is shown in Figure 2.21.

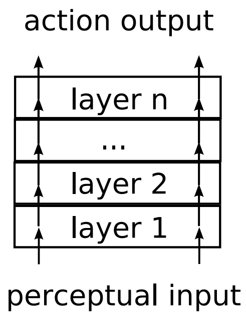


Figure 2.21.: Information and control flow of a vertical one pass layered architecture; [MPT95, p. 263]

The two pass vertical layered architecture pass processed input up to higher layers in the architecture, where more abstracted decision making is implemented. The information and control flow in such an architecture is sketched in Figure 2.22. The results of higher layers are delegated back to the lower layers for execution. A prominent example of systems based on layered architectures are the InterRRap architecture presented, e.g., by Fischer et al. [FMP96, pp. 403–405].

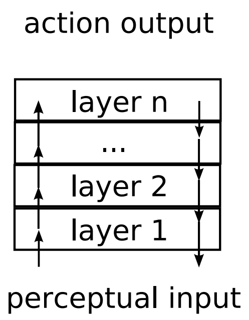


Figure 2.22.: Information and control flow of a vertical two pass layered architecture; [MPT95, p. 263]

Layered architectures have also been established in robotics, e.g., Alamai et al. [ACF$^+$98]. The authors present an architecture where aspects like collision-avoidance are implemented based on a reflex-based layer while more complex steering or planning activities are done on higher-levels of abstraction. Another example of a layered architecture in robotics is the architecture of Stanley the autonomous driving robot, e.g., Wooldridge [Woo09, p. 99]. A detailed description of Stanley's architecture can be found in Thrun et al. [TMD$^+$06].

These architectures remain abstract and have to be instantiated based on the existing agent implementation languages or frameworks.

2.3.3. Internal state representation

In the following the structure of the internal state of an agent is presented. Therefore, we review the concept of the state of an agent in the aforementioned architectures.

The state of a logic-based agent is the sum of formulae that are in his database. These formulae encodes what the agent assumes to be true about its environment. If reactive agents use an explicit state model, like model-based reflex agents (introduced in Section 2.3.1) their internal state is the current state they are currently in. Reflex agents without the notion of states, like simple reflex agents, can be modeled as stateful agents with exactly one state. For example, if agent a has the set of possible internal states S, at time t its internal state is $s_t \in S$.

The state of a BDI agent is given by its current beliefs, desires and intentions (B,D,I) [Woo99, p. 58]. With B$\subseteq$ *Bel*, whereby *Bel* the set of all possible beliefs is, D $\subseteq$ *Des* with *Des* the set of all possible desires, and I $\subseteq$ *Int* with *Int* the set of all possible intentions.

The state of an agent with a layered architecture is harder to specify. From an abstract perspective it is, like in the case of reactive agent defined as an element of the powerset of states each layer can have. But the states of a layer cannot be enumerated on this abstract level, as this depends on what layers exist and how they are realized. Nevertheless, it is possible to specify their state, as, for instance, discussed by Thrun et al. [TMD+06, p. 666].

Even if no specific architecture is referred to, there exist common understanding of some aspects of the agent's internal state, which is also referred to as belief set , e.g., by Decker [Dec95], or knowledge-base, e.g., by Finin et al. [FFMM93]. The term knowledge-base origins from research done in knowledge-based systems. The term belief set is motivated by the aforementioned BDI architecture, although it is not limited to those architectures. Russell and Norvig use the term belief set in the following notion:

> "We call each such set of states a belief state, representing the agent's current beliefs about the possible physical state it might be in." [RN03, p. 84]

Thus, the term belief set is used to refer to the representation of the environment the agent has perceived, so far. This notation does not have any implication for the architecture of an agent. There are two way how a fact can be added to the belief set, either by perception of the agent, including communication with other agents, or by reasoning, done by the agent itself. Therefore, the agent has some form of logics that enables him to add new facts, derived from existing ones to the internal state.

Thus, theoretically the knowledge of such an agent at a certain point in time is defined by its logics and its internal state. From a more practical perspective it has to be mentioned that the available computational power of that agents is a limiting factor. If a fact is not in its internal state and can be derived from it, this fact is unknown to the agent.

2.3.4. Multiagent system: foundation for emergence

If multiple agents form one system, this system can be called a *multiagent System* (MAS). A key characteristic of a MAS is that the agents within a MAS can interact via direct or indirect communication.

In previous research, e.g., von Martial [Mar92], Ossowski [Oss98] or Rosenschein and Zlotkin [RZ98a] systems with multiple agents has been distinct in two subclasses: distributed problem solving systems and multiagent systems.

Distributed problem solving systems are systems composed out of different agents, that have been designed to solve a given problem, most probably by one designer. The entire system aims to solve a global problem and thus, the agents in such a system are benevolent. In contrast a MAS was composed out of agents that only try to optimize their local utility function that was given by their designer. So in one system agents from different designers and with different objective functions can exist. The entire system does not necessarily has to have a global goal that has to be achieved. A more detailed discussion is provided by Weiss [Wei99, p. 4] or by Jennings et al. [JSW98].

This distinction has become less relevant as a lot of researchers investigated hybrid forms of those systems. The term multiagent system is now widely accepted as a system of multiple agents, independently of their local utility functions. So according to Wooldridge it can be defined as follows:

> "Multiagent systems are systems composed of multiple interaction computing elements, known as *agents*." [Woo09, p. xiii]

An even stricter version of this definition is given by Weiss, where the agents have to be intelligent[13].

> [A multiagent system is a] "...systens in which several interaction, intelligent agents pursue some set of goals of perform some set of tasks." [Wei99, p. 1]

It has to be mentioned that these definitions are not exclusive, there exist other definitions as well. But they were chosen by the author as they offer a valid foundation for this study, and they are widely accept in the research community.

According to Jennings et al. [JSW98] a MAS can be characterized by the facts that

- each agent has incomplete information, or capabilities for solving the problem, thus each agent has a limited viewpoint;
- there is no global system control;
- data is decentralized; and
- computation is asynchronous.

[13]Thereby, flexibility and rationality are pointed out as key characteristics of those intelligent agents [Wei99, pp. 2,3].

As a consequence of these characteristics presented by Jennings et al. a MAS can show emergence behavior as it has been characterized by Timm [Tim04, p. 24]. According to Timm emergence behavior is

- surprising, as it cannot be predicted as a consequence of simple interactions,
- establishing organization forms in complex systems that have not been intended before, and
- creating intelligent behavior, based on the interactions of individuals on a less intelligent level.

Communication in MAS

As pointed out before, the interaction among agents plays a central role in MAS. Thereby, communication play a central role, as the most prominent form of agent interaction. We present here the foundations of agent communication because communication is essential for coordination or by quoting Ferber "all coordination must necessarily go through communication" [Fer99, p. 407].

Agent interaction can be direct or indirect. Indirect interaction uses parts of a shared environment as a mediator. Indirect interaction can either be intended as indirect communication or can be coincidence as two agents interfere as they access a non-sharable resource in the environment. Indirect communication can be implemented by marking the environment, e.g., with artificial pheromone traces which are used in ant algorithms.

Direct interaction between agents is done via communication, i.e., message passing. This communication requires a common language, a shared ontology and an agent communication infrastructure like a yellow-page service and a message forwarding protocol. Major research efforts have been undertaken to build appropriate forms of agent communication languages (ACL), knowledge representation and communication infrastructure. These efforts included the *knowledge query and manipulation language* (KQML) and the *knowledge interchange format* (KIF) (see Huhns and Stephens [HS99] or Wooldridge [Woo09, Chap. 7] for an introduction). The appropriate design of ontologies for agent communication is described by Wooldridge [Woo09, Chap. 6] and a good description for ontology design in general is provided by Noy and McGuinness [NM01]. Both KIF and KQML were developed within the DARPA knowledge sharing effort (summarized by Finin et al. [FFMM93], for instance, and were forerunners of the FIPA standards. FIPA is the *Foundation for Intelligent Physical Agents* an IEEE standards organization. More details about the FIPA organization can be found at the FIPA homepage [FIP10][14]. FIPA has standardized among other:

[14]`http://www.fipa.org/index.html` accessible at 02/04/2010

Parameter	**Category of Parameters**
performative	Type of communicative acts
sender	Participant in communication
receiver	Participant in communication
reply-to	Participant in communication
content	Content of message
language	Description of Content
encoding	Description of Content
ontology	Description of Content
protocol	Control of conversation
conversation-id	Control of conversation
reply-with	Control of conversation
in-reply-to	Control of conversation
reply-by	Control of conversation

Table 2.5.: FIPA ACL Message Parameters [FIP02c, p. 2]

- agents and agent platforms, e.g., [FIP02a, FIP02h]
- Message Transportation, e.g., [FIP02d, FIP02b]
- Message structure, e.g., [FIP02n, FIP02c]
- interaction protocols[15], e.g., [FIP02l, FIP02j]

All standards can be downloaded for free from the referred websites. An overview of the standards can be found as well[16]. Note that within the FIPA standards only direct communication is considered [FIP02a, pp. 14–16]. The communication facilities have to be provided by the agent environment see Huhns and Stephens [HS99, p. 81] and FIPA standard [FIP02a].

According to FIPA the exchange of messages between agents it the favorable form for inter-agent communication. To ensure efficient communication and interoperability the format of messages is standardized by FIPA. The overall structure of the message format has been outlined in FIPA standard [FIP02a, pp. 28–35]. In standard [FIP02c] is has been detailed. A message contains a number of fields, also called parameters. An overview of these fields is shown in Table 2.5. This format has been influenced by previously research concerning KQML. An introduction to KQML and FIPA based communication can be found in Wooldridge [Woo09, pp. 136–146].

[15]FIPA interaction protocols are discussed in more detail below.

[16]`http://www.fipa.org/repository/standardspecs.html` Accessed: 02/04/2010

Interaction protocol	**relation quantity**
Request Interaction Protocol [FIP02l]	1:1
Query Interaction Protocol [FIP02j]	1:1
Request When Interaction Protocol [FIP02m]	1:1
Contract Net Interaction Protocol [FIP02f]	1:n
Iterated Contract Net Interaction Protocol [FIP02g]	1:n
Brokering Interaction Protocol [FIP02e]	1:1
Recruiting Interaction Protocol [FIP02k]	1:1
Subscribe Interaction Protocol [FIP02o]	1:1
Propose Interaction Protocol [FIP02i]	1:1
English Auction Interaction Protocol [FIP01c]	1:n
Dutch Auction Interaction Protocol [FIP01b]	1:n

Table 2.6.: FIPA interaction protocols and their cardinality

Interaction protocols among agents play an important role, as they structure a sequence of messages that are exchanged between agents. As Huhns and Stephens point out by means of these interaction protocols agents can have a conversation

> "Interaction protocols govern the exchange of a series of messages among agents - a conversation." [HS99, p. 96] [17]

Interaction protocols can be defined for one sender and one receiver, than they are called *binary* as they represent a 1:1 relationship. Although interaction protocols can be defined for one sender and n receivers, in which case they are called *n-ary* as they represent a 1:n relationship [HS99, p. 86]. Wooldridge [Woo09, p. 317] also mention the case of many-to-many negotiations. But this type of negotiations is rare to find in multiagent research, probably because of the resulting complexity. For the interaction protocols specified by the FIPA these cardinalities of the interaction protocols are shown in Table 2.6.

Note that the Dutch and English auction protocol are no confirmed standards by the FIPA yet. They are still in the experimental stage, but shown here for completeness of the specified FIPA interaction protocols.

In the FIPA standards the interaction protocols are visualized by an UML1.x extension named *Agent UML* (AUML) presented by Odell et al. [OPB01]. AUML is discussed in more detail in Section 3.2.1 in the context of agent-oriented software engineering.

[17] A similar statement can be found at [HS99, p. 79].

3. State of the art

In this chapter we give an introduction to the current state of the art of the research of coordination in multiagent systems. Nevertheless, the field of coordination is not a exclusive for researchers from the field of DAI. For that reason research concerning coordination from related fields, like business administration (Section 3.1.1) and game theory (Section 3.1.3) are discussed in this chapter, as well.

In the second section of this chapter the engineering of DAI systems, typically termed agent-oriented software engineering (AOSE) is focused and especially aspects of research concerning reuse of concepts in AOSE is presented. Thereby, we point out the current shortcomings in the field of reuse of methodologies in AOSE and for the coordination of planning systems in particular. Based on this findings we identify the research question and issues this thesis has to tackle.

3.1. Coordination

Research considering coordination has attracted researchers from different fields of research and thus the body of existing work is hardly to survey, at all. And even within the lines of research, like computer science. The scope of research concerning coordination is wide. Omicini et al. [ORVR04] consequently paraphrase it in the following way:

> "Research on coordination possibly represents the most un-coordinated activity in the history of computer science." [ORVR04]

Consequently, in this section only an extraction of existing work is presented, from fields that have a direct connection to DAI, and thus it cannot claim to be complete. In the following coordination is described from a rather broad perspective and in the following subsections focused from the perspective of the fields of business administration, DAI, and game theory.

The term coordination is used in different disciplines with different notations. Coordination as a topic by itself can only be found only rarely. One of the most prominent papers that suggests to establish a "coordination theory" has been presented by Malone and Crowston [MC94, p. 88]. In their interdisciplinary study they compare the research about coordination in different fields of research,

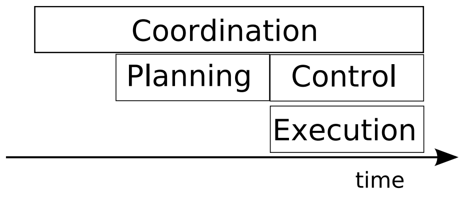


Figure 3.1.: Timeline of coordination activities

in particular they address the fields of computer science, economics, operations research, and organization theory. A consequence of their work is the widely accepted definition of coordination. In their work they define coordination as:

> "Coordination is managing dependencies between activities." [MC94, p. 90]

A fundamental underlying problem of coordination is that actions and/or decisions are executed/made in a distributed manner. And as Malone and Crowston pointed out the decision making and/or execution have to be managed to achieve a goal or perform an efficient behavior [MC94, 90]. Thus, in all coordination scenarios we can find a distributed scenario, in other cases all decision making and execution are centralized and no coordination is necessary. Then the coordination problem becomes a planning problem.

Nevertheless, the term coordination comprises a wide field of activities, as it does not point out when the coordination takes place. This is pointed out in the Figure 3.1. Activities are executed in the execution phase. The execution of activities is supervised by a controlling entity. Before the execution of a set of actions is started, typically a planning process has arranged the activities. Coordination can be performed during execution/control, planning, or even before the planning phase started.

A group of research addresses coordination from strategic [WCDH00] and operational [CWDH00] point of view. The strategic coordination is not addressed in this study, as already pointed out. The work done by Coates et al. [CWDH00] is more an update of the work done by Malone and Crowston addressing resource management, planning, and scheduling. The authors surveyed the application of coordination techniques in different fields of application. Among other they present planning and scheduling as techniques for coordination.

The vision of Malone and Crowston [MC94] that a research field for *coordination* forms has not become truth so far. Today coordination is subject of research in different areas. Therefore, it is fruitful to address current fields of research dealing with coordination. Another introduction into the field of research concerning coordination from the DAI perspective is provided by Ossowski [Oss98, Chap. 2].

3.1.1. Coordination concepts in business administration

In this section the perspective of business administration on coordination is presented. Thereby, different perspectives are presented. According to Hansen and Neumann a number of theories exist that try to explain the coordination of economical exchange processes [HN09, p. 945]. These authors mention, for instance, the exchange theory[1] and the resource dependence theory. The exchange theory is rooted in sociology. It assumes that resources are rare, the entities/participants have different specializations/abilities. Under these assumption an exchange happens between the participants if they both gain profit from the exchange. The main idea of the resource dependence theory is that companies are dependent on exchange facing rare resources. A consequence is that the companies will lose autonomy as they depend on external resources. According to this theory this leads to establishing networks, to make the dependency bilateral. Companies engage in networks to reduce their dependency and gain influence on other [CG01, pp. 14–15].

The approaches and ideas presented in this section form the context and administrative background for the development of actual information and planning systems that are established today.

In the following the notion of coordination is presented from the perspective of decision-oriented business administration[2]. This field of research has been established by Heinen [Hei86]. A practically motivated introduction is presented by Adam et al. [ABTV04]. The focus of this line of research lies on the decision making [ABTV04, p. 1]. Decisions are the consequences of plans, developed for models of the reality [ABTV04, p. 3]. Thus, decision making is a consequence of planning activities. Thereby, the planning problem is decomposed into four elements [ABTV04, p. 3]:

1. decision field element: containing all causal relationships, necessary to estimate the effects of different action alternatives.
2. evaluation element: mapping the expected outcome to an economic quality measure.
3. objective element: describes what should be achieved.
4. decision making element: computes the optimal solution

If all these aspects are well defined, i.e., they are fully operational defined, then decision making becomes a computing task [ABTV04, p. 4]. In fact the

[1]in German: Austauschtheorie
[2]in German originally termed “Entscheidungsorientierte Betriebswirtschaftslehre”

authors state that these problems can be rarely found in real world settings. Most often the planning problem definition has some defects that prohibit an effective computation. Possible defects are [ABTV04, p. 4,5]:

- Causal defects: the causal relationships cannot be described in detail.
- Evaluation defects: the results of actions cannot directly mapped to a one dimensional economic metric.
- Objective defects: the objective cannot be formulated precisely in a one dimensional objective function.
- Solution defects: there exists no efficient method for solving the planning problem.

Those defects are, from the business administration point of view, the motivation for dividing defect problems into a number of less defect sub-problems. Even though the authors are aware of the fact that this divide and conquer strategy can endanger the overall solution quality[3].

The decomposition of planning problems can be done along two different dimensions [Wöh02, pp. 104–108]:

- factually, e.g., distribution, finance, production, and
- chronological, e.g., in strategically, tactically and operationally planning.

Consequently, resulting interdependencies have factually or chronological characteristics. Corsten and Gössinger summarize those dependencies among other as objective interdependencies[4] [CG01, pp. 45–47]. The taxonomy of Corsten and Gössinger is presented in Figure 3.2. According to the taxonomy of interdependencies presented by these authors objective interdependencies can exist either by restrictions or goals. A restriction is defined by physical restriction imposing the interdependencies. Shared resources or precedence constraint during a production are example for such dependencies. A goal dependency can be defined in form of profit, evaluation measurements or risk dependencies on other entities. In contrast to objective interdependencies are behavior interdependencies. A behavior interdependency exists if the decision making of an entity depends on, or is taken into account, the expected decision making behavior of another entity. This is, for instance, a typically assumption in game theory, when there exists no dominate strategy.

[3]In German “Die Bildung von Teilprobleme fördert Partialsichten und steht daher einem übergreifenden Denken entgegen.” [ABTV04, p. 8].

[4]In German “Sachinterdependenzen”, to avoid ambiguities we translated them not as factual interdependencies.

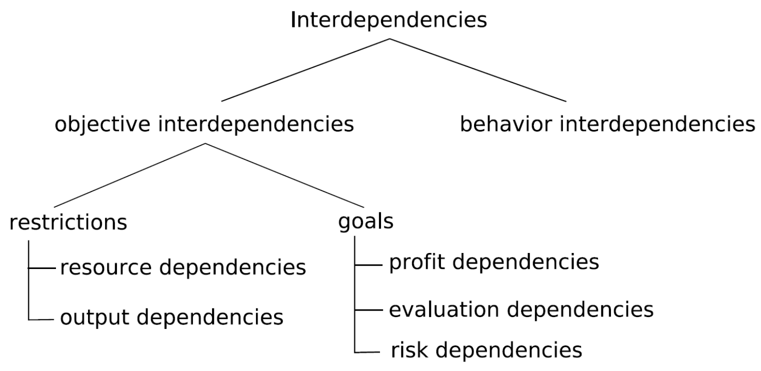


Figure 3.2.: Taxonomy of interdependencies adapted from [CG01, p. 47]

A typical example for the decomposition of large problems in business administration is the management of a company. According to Adam et al. [ABTV04, p. 7] and Wöhe [Wöh02, p. 332] management of a company can be decomposed into the sub-problems of:

- distribution planning
- production planning
- investment planning
- finance planning

This is in fact a factually decomposition from a strategic perspective. Each of these planning problems is abstract and complex, such that automation on this level of detail is not realistic. Especially as the planning has to respect various only weakly defined constraints, like aspects from legal regulations, corporate policies and secondary objectives.

A consequence of the decomposition is the necessity for coordination. During the integration of the solution of sub-problems, existing dependencies have to respected [ABTV04, p. 5]. This arises due cutting of potentially existing dependencies in the decomposition step. Therefore, Adam et al. characterize coordination as adjusting and controlling dependent decision fields from an overall perspective[5]. The authors have an abstract view on coordination, which becomes obvious in their classification of coordination techniques that is shown in Figure 3.3. According to their classification, coordination can either be done by market-

[5] "Die Notwendigkeit zur Koordination ergibt sich aus der Arbeitsteilung bzw. Teilproblembildung, wenn durch die Zerlegung die zwischen den gebildeten Teilbereichen bestehende Kopplungen zerschnitten werden." [ABTV04, p .10]

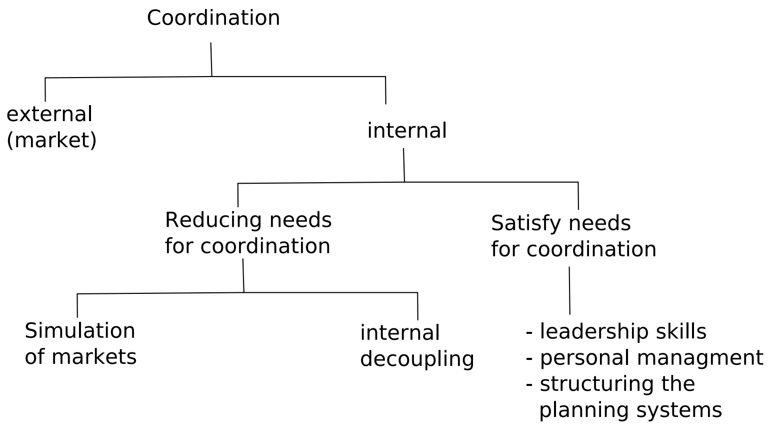


Figure 3.3.: Types of coordination, according to [ABTV04, p. 21]

based mechanisms or by mechanisms relying on hierarchy, which is also stated by Hansen and Neumann [HN09, p. 946]. These authors agree that hybrid approaches are possible, too. Examples for hybrid coordination forms are, for instance, enterprise networks [HN09, p. 946]. Enterprise networks have been extensively studies, e.g., by Corsten and Gössinger [CG01].

The authors discuss that the selection of the appropriate coordination approach on this coarse grained level depends on the environment[6]. If the environment is simple, static and has a low degree of uncertainty then centralized hierarchical coordination promises to be more efficient. While in more complex, dynamic environments with possibly higher degree of uncertainty the coordination has to be appropriate to the current situation. This can be achieved by decentralized and self-aware decision making [ABTV04, pp .23]. Thus, changing environment enforces paradigm shifts in the coordination philosophy [ABTV04, p. 24].

Taking the assumption that coordination deals mainly with the integration of different plans that have been motivated so far. We now discuss more concrete techniques that have been proposed in business administration. Wöhe [Wöh02, pp. 108–112] presents three forms how planning could be structured:

- continuous planning for the combination of chronological decomposed planning,
- top-down, bottom-up or top-down/bottom-up planning for the combination of different partial plans, or
- bottleneck-first planning.

[6]This statement is true, also for more concrete coordination mechanisms, see [Dec95].

Continuous planning uses different levels of details for planning ahead of time. The nearby future is planned very accurate, while parts of the plan that are further ahead in time are only planned based on rough estimations or on a coarse grained level. Finding the right level of detail can become a problem [Kur05, p. 40].

In the following the top-down, bottom-up and the top-down/bottom-up approach are presented and references to some applications of coordination in these ways are given. More details can be found at [Wöh02, pp. 109–110].

The top-down approach for combining existing partial plans, assumes that there exist a superior entity that create abstract plans that are refined in subordinated entities. Thereby, plans become more and more concrete. To apply this approach it is necessary that each superordinate entity has enough information about the capabilities of all of its direct subordinated entities. This enables the superordinate entity to generate feasible abstract plans that can be implemented by all its subordinates. This problem is referred to as vertical interdependencies. This implies that the information has to be available to each superordinate entity and it has to be updated regularly. The information availability and quality is critical, thus it is mentioned by Kurbel as a counter-argument for the use of simultaneous planning [Kur05, p. 40]. The idea of top-down planning has been implemented, for instance, in hierarchical planning [HM73]. This approach have similarities with the approach to perform coordination in a pre-planning phase, which is proposed by Witteveen et al. [WW06a, SW07]. The authors propose to add additional constraints to the original planning problem to ensure that the local planning does not generate conflicting plans.

The bottom-up approach for combining existing partial plans follows a different approach. Each entity plans its activities locally. To enforce a feasible plan all these partial plans are communicated to a coordinating entity that is superordinate to all planning entities. It is the task of this entity to ensure that the resulting partial plans are coordinated. This approach assumes that the coordinating entity has at least abstracted knowledge of the capabilities of the local planning entities. Moreover, it requires some coordination knowledge. The coordination knowledge comprises knowledge how conflicts in different partial plans are can be resolved. In a worst case scenario, were all partial plans are maximal conflicting the coordination entity has the task to implement a simultaneous planning. An example of such a post-planning coordination step has been proposed for instance by von Martial [Mar92].

In the top down/bottom up approach is a first step guidelines are generated in a top down way. The feasibility of these abstract plans is then tested by the subordinate planning entities that try to generate a concrete plan respecting all given guidelines. If a conflict occurs or if it is not possible to fulfill all guidelines a feedback is given and the superordinate entity has to adapt its plan. The incorporation of feedback in the planning give the global planning entity additional information about the local abilities to generate feasible plans and that also can be used to adapt the global plan to changes that may occur during execution. An example of such a planning system for multi-site scheduling has been proposed within the MUST approach by Sauer [Sau02].

A heuristic approach to handle the coordination problem is, as described earlier, the sequential solution of the partial problems, and the usage of the results of earlier computations in later ones [Wöh02, p. 331]. Thereby, the ordering of the partial problems should be done in a bottleneck-first heuristic using forward and backward planning from the bottleneck [Wöh02, p. 331]. Of course, this approach suffers if multiple bottlenecks exist or bottlenecks depends on the current workload. Moreover, in reactive planning/scheduling scenarios the bottlenecks can shift depending on the event that occurred. The interdependencies of planning systems cannot be handled in this approach, too.

Summarizing, there exists literature that has analyzed the motivation for the dividing of decision/planning problems and the required integration of partial solutions. Nevertheless, the findings concerning coordination remain abstract, from a computer scientists point of view, as these models cannot be operationalized easily.

Another aspect that is mentioned, e.g., by Kurbel [Kur05, pp. 244–246] and Hansen and Neumann [HN09, p. 947] is the efficient, ideally automated, information flow between different entities. Thus, it is emphasized in business informatics that electronic data exchange [HN09, pp. 958–965] and enterprise application integration [RMB01] are important technologies to enable efficient coordination. These approaches mainly deal with the required information infrastructure needed for the management to coordinate the actions of different companies not only on the strategic level.

Recently, there exists although a trend in business administration to apply technologies that have been developed in the filed of DAI as well as joint research efforts. For instance, Meier [Mei08] deals in his work with the coordination of different planning systems in the container terminal management problem, see the motivating example, Section 1.2. Coordination is achieved by wrapping the planning systems with agents and communicates the plans, or more precisely specific relevant information for planning, to agents who are responsible for plan/planning system that depend on the first one.

In the context of a joint research project, conducted by economics and computer scientist coordination techniques incorporating multiagent systems have been investigated as well. In the priority research program “Intelligent Agents and Realistic Commercial Application Scenarios” Agent.Enterprise has been introduced as demonstrator for distributed planning and scheduling. An introduction can be found in Woelk et al. [WRZN06]. In the context of this effort mechanisms for coordinating companies forming a supply chain has been investigated. Results have been presented in Frey et al. [FSWZ03] and Stockheim et al.[SSWG02], for instance.

The field of auctions has been excluded from this section as they are discussed in the section coordination in game theory and mechanism design (see Section 3.1.3).

Collaborative planning in operations research

The need for coordination among different autonomous planning systems has also been realized in the field of operations research. This research area has mainly evolved from the field of supply chain management [Sta09], as a consequence of the fact that conventional production planning techniques like MRP II (for an introduction see [Kur05, Chap. 4]) are not suitable for supply chains [Dud04, p. 14]. According to the perspective of operations research there exists two approaches how plans can be coordinated in a supply chain, either by hierarchical or collaborative planning [Dud04, p. 12]. Hierarchical planning has been mentioned above and is implemented in current *advanced planning system* solutions, see Dudeck [Dud04, p. 15], and Stadler and Kilger [SK02]. According to Stadtler [Sta09, p. 5] hierarchical planning approaches suffer, as already mention, from the inherent information asymmetry in networks of autonomous entities. Moreover, Stadtler points out that hierarchical planning cannot cope with the existence of local, possibly conflicting, objective functions. Each planning entity might have local constraints that might cause conflicts [Sta09, p. 5]. In contrast to hierarchical planning collaborative planning aims to overcome these weaknesses by accepting the fact that autonomous entities have to coordinate their activities. According to a broad definition collaborative planning can be defined

> “as a joint decision making process for aligning plans of individual SC [supply chain] members with the aim of achieving coordination in light of information asymmetry.” [Sta09, p. 6]

A typical collaborative planning mechanism has been presented by Dudeck [Dud04]. He present a collaboration planning mechanism based on non-hierarchical negotiation among independent supply chain partners. Doing so, the master plans have been identified as the appropriate level for collaborative planning [Dud04, p. 20]. Note that a master plan frames the production for a time horizon for about 3 up to 18 month ahead, depending on the line of business. Dudeck assumes that all supply chain partners apply a mathematical program to solve their local planning problem. As a local planning problem the multi-level capacitated lot-sizing problem [Dud04, pp. 30–32] is chosen. Planning entities exchange proposals and counterproposals that have been generated by their local planning systems. But this approach is limited by the fact that a mathematical program is assumed for all supply chain partners and that the coordination focuses primarily on the material supply on a coarse grained level, which are rather typically restrictions for approaches presented in the field of collaborative planning.

Techniques used in collaborative planning rely on the publication of information concerning local cost structures or side payments. Within a side payment an agent gives or asks for additional payment to commit to a change in its plan, e.g., by adding an additional task. Both techniques can be used to achieve solutions of higher quality from the global perspective. But these techniques impose the risk of agents behave opportunistic and thus do not publish information truthfully. These risks have been discussed, e.g., by Dudeck [Dud04, Chap. 6.2].

As already pointed out most of the work on collaborative planning has been performed on rather abstract plans, like master plans, that fixes the activities for the next couples of months. A work concerning more concrete schedules have been presented by Scheckenbach [Sch09]. In his thesis, Scheckenbach assumes that at each side a resource constraint project scheduling problem has to be solved, which is solved by an SAP Detailed Scheduling Optimizer [Sch09, pp. 59, 87]. The plans are much more detailed. Even though a restricting assumption is used, that all planning entities have to solve homogenous planning problems. Interestingly Scheckenbach does not use side-payments, as he argue that non-carefully designed side-payment schemes leads to agents that cheat and probe. For coordination Scheckenbach propose a mixture of hierarchical planning and a distributed evolutionary algorithm. More details concerning his methodology can be found in [Sch09, Chap. 4].

A work concerning collaborative planning that is not based in the application field of supply chain management, has been presented by Püttmann [Püt07]. Püttmann addresses the coordination within a transportation network, concerning the transport of goods by multiple parties, in intermodal freight transportation. This scenario address a coordination that is comes closer to the operational level, than the coordination in a supply chain. Their approach base on result-sharing techniques and is based on a mathematical optimization model, presented in Püttmann and Stadtler [PS10].

A recent survey of the state-of-the-art has been presented by Stadtler [Sta09]. Stadtler also suggest a framework for classifying these coordination approaches according to three categories [Sta09, p. 12]:

- structure of the supply chain and it's members,

- the decision structure of each supply chain member, and
- characteristics of the collaboration planning scheme.

The first two characteristics describe properties of the problem domain, the third one tries to gather properties of solutions. These categories are not discussed here in more detail.

Between the field of collaborative planning and multiagent systems research a number of similar assumptions and metaphors exist: for example decision makers are distributed and are autonomously, their interaction is specified by a protocol [Sta09, p. 20], and negotiations are used to coordinate their plans. Even though similarities exist there has not been much exchange between those fields of research. From the perspective of collaborative planning software agents are seen as one implementation architectures for collaborative planning mechanisms. They are primarily seen as a pure software architectural issue [Sta09, p. 11].

Negotiation in business administration

A common technique to reach coordinated behavior are negotiations. Negotiations can be modeled in different fields, and are discussed in subsequent sections especially from the perspective of DAI. In the field of business administration a negotiation can be defined as:

> “A negotiation is an interactive communication process that may take place whenever we want something from someone else or another person wants something from us.” [She06, p. 6]

There exist no protocols or constraints on the interaction. The negotiation process can be characterized by four steps:

- preparation,
- information exchange,
- bargaining, and
- closing, commitment.

During a negotiation the negotiator should therefore search for boundaries and options for possible agreements that allow reciprocal accommodation, because a deal can only be reached in a win-win situation. Shell [She06, p. xvi] argues that all deals made between human negotiators have to be win-win situation because otherwise there would not exist a deal, or it would be irrational for for one party to participate in the deal. Moreover, the win-win situation is a strong motivation for the implementation of the deal. So if both sides profits from the deal, no one has a motivation to break or resolve the deal, as long as their exist no more profitable solution. Negotiations from the perspective of business adminstration have a significant overlap with psychology. For instance, Shell points out that it is important that "people need to feel they have earned concessions" [She06, p. xviii]. Regardless of the fact if these concessions are real or would have been given without negotiations anyway. Moreover, ethical guidelines and personal negotiation styles, as part of an individual character of the negotiator play's a significant role. Shell distinct between the following negotiation styles [She06, pp. 9–12]:

- avoiding,
- compromise,
- accommodation,
- competitive, and
- collaborative/problem solving.

Even though a lot of psychological aspects come into play, it should not be the dominant factor in a negotiation. But those are aspects that have to be taken into account in the preparation phase of a negotiation. Therefore, authors like Fisher et al. [FUP91] suggest that the negotiator should try to keep out these personal feelings and focus on the problem and not the people. Moreover, in human negotiations it has to be mentioned that the motivation for people to negotiate is an important information, as they try to achieve a certain interest that is not necessarily identical to their initial position. But this information can be crucial to come to a deal that allow all negotiating parties to gain something out of the negotiation and therefore accommodate all parts. Fisher et al. [FUP91] therefore present the idea of the principled negotiation strategy, that tries to focus on the problem at hand, and avoid some of the inter-personal psychology aspects that can hinders finding a solution during a negotiation.

Although in the field of business administration the idea that negotiations can be automated by agents representing humans. These automated negotiations should be inspired and guided by principles discovered in negotiations done by humans. This idea has been adopted by researchers in multiagent systems as well. So Vassileva and Mudgal [VM02] argue that it is important to ensure a win-win situation. And in Schumann, Kurtanovic and Timm [SKT09] a methodology to model negotiation strategies and trade-offs in a human readable way, that can automatically transformed to information accessible for agents, is presented.

3.1.2. Coordination in DAI

In this section research concerning coordination in the field of DAI is addressed. In general coordination in multiagent systems research tries to impose a desired behavior among autonomous agents. Note that we try to limit on research primarily rooted in DAI, research closely related to DAI rooted in game theory, like the research on decommitments done by Sandholm, is presented in Section 3.1.3. The approaches are grouped into three different fields of research. First, research based on classical AI approaches is presented. The second line of research is strongly motivated by research from distributed computing and resulting coordination mechanisms. Finally, in the third section the approaches combining the previous ones are discussed.

Classical DAI coordination

At first, the modeling of dependencies performed in AI research is presented. Thereafter mechanisms for handling existing dependencies are described. These mechanisms base on ideas of task or result sharing techniques. Finally, coordination is considered as a planning problem and appropriate techniques are discussed.

Modeling and detecting of dependencies According to Ossowski [Oss08] the coordination process can be decomposed into two parts. First, existing dependencies have to be detected, and second, coordination actions have to be selected that are appropriate. Both are formulated within a *coordination mechanism* that an agent can apply. But as Omicini et al. [ORVR04, p. 275] state dependency detection is a knowledge intensive task, that can hardly be achieved by agents based on local knowledge and local perception. As dependency detection is a topic on its own, it is not detailed in this thesis.

Taxonomy of dependencies The most widespread taxonomy of dependencies, that can exist between different plans of agents, has been presented by von Martial [Mar92]. This taxonomy is shown in Figure 3.4. Dependencies can either be negative or positive. Negative relationships between plans can be resource conflicts concerning some consumable or non-consumable resources. A conflict concerning a consumable resource is characterized by a demand concerning this resource that exceeds the existing amount of this resource. A conflict concerning a non-consumable resource is given if two or more agents are not allowed to use the same resource at the same time. Other negative relationships are called incompatibilities between plans. An example for such a plan incompatibility is a situation in which two agents who wants to meet and end up at different places, as they did not coordinate properly on the locality to meet.

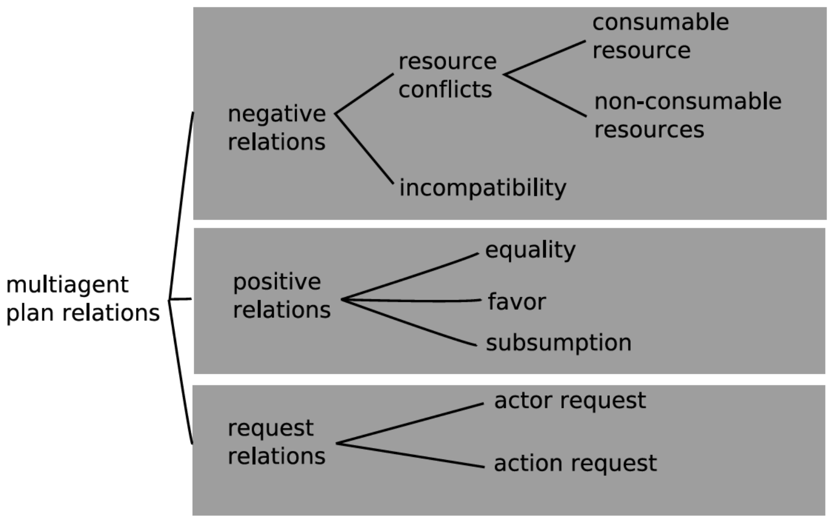


Figure 3.4.: Taxonomy of plan relations according to [Mar92, p. 90]

Positive relationships are equality, favor, and subsumption. An equality relation exists in two plans, if there is redundancy between the plans that could be avoided if only one of the agents performs the task. A favor relationship exists between if the plan of one agent can contribute to the execution of another agent's plan. A subsumption relation exists if an action in a plan of an agent will lead to the fulfillment of a goal in another agent's plan. Thus, the second agent is not required to incorporate the necessary actions to achieve this goal in its plan, as a more general action has already been planned.

The third class of relations that can exist are the *obvious* relations between two plans of agents. They are obvious, as they are requested by the agent. An agent can either perform an actor request or an action request. Using an actor request an agent requests a specific agent to perform an action. In contrast in an action request an agent requests that an action is performed, regardless by which agent the action is performed.

Another classification of dependencies has been proposed by Malone and Crowston [MC94]. This classification origins from their attempt of the authors to survey different fields of research like organization theory, computer science and economics for coordination techniques. A simplified version of their taxonomy is presented in Table 3.1. Note that the authors do not claim that this structure is complete. Malone and Crowston identified four different types of dependencies, that can have special subcategories [MC94, pp. 91–97]. The first category are shared resources, where different activities share limited resources, in those cases

Dependency	**Example for coordination process**
shared resources	e.g., *first come/first serve*, priority order
task assignment	as shared resources
producer/consumer relationships	
prerequisite constraint	notification, sequencing, tracking
transfer	inventory management
usability	standardization, participatory design
simultaneity constraints	scheduling, synchronization
task/subtask	goal selection, task decomposition

Table 3.1.: Dependency classification, simplified version of table in [MC94, p. 91]

a resource allocation has to be done, to coordinate the activities. Task assignment is seen as a special case of resource allocation, where the rare resource is the time of the actors.

The second large group of dependencies are producer/consumer relationships. Malone and Crowston identified different subcategories. Prerequisite constraints ensure that an activity is finished before the consumer's activity starts. Activities have to be sequenced or a notification mechanism has to be in place. The second subcategory is transfer. Thereby, it is indicated that a *product* that has been generated by the producer must be delivered/transported to the consumer as well. This can be done either by physical transportation of goods or by communication of results. The third subcategory is usability. This aspect is mainly motivated by the design of goods, but by far means limited to such scenarios. The *product* has to be usable for the consumer. This implies that it has all needed properties to get used by the consumer and that the consumer is able to use it. Example for techniques to coordinate these dependencies are data-exchange standard that define the syntax and semantic of information that is send via communication.

The third category of dependencies are simultaneity constraint, so that two or more activities have to be performed simultaneously. An example where such dependencies have to be dealt with is meeting scheduling or other conventional synchronization problems in computer science, where often things are not allowed to happen simultaneously.

The fourth dependency category are task/subtask relationships. Tasks are refined by subtask. Thus, to fulfill a task it is typically necessary to fulfill all is to be assumed if its subtasks are fulfilled. Coordination approaches like task decomposition or hierarchical planning might be used to coordinate these dependencies.

Other ways to classify dependencies have been proposed by Jennings [Jen96] and Decker [Dec95]. A dependency that has to be respected is called *strong depen-*

dencies [Jen96, p. 192] or *hard coordination relationships* [Dec95, p. 136]. These relations have the character of negative relationships according to von Martials classification. Opposed to these strict dependencies *weak dependencies* [Jen96, p. 192] or *soft coordination relationships* [Dec95, p. 136] are dependencies that do not necessarily have to be regarded. Of course, all positive relationships have such a character. But not all weak dependencies are positive for the overall plans. Decker, for example, classifies its *hinders* relation (see the TÆMS description below) as a soft constraint relationship [Dec95, p. 136]. It goes without saying that dependencies between two entities are directed. Thus, dependencies between two plans can be either unidirectional, i.e., only one plan depends on the other; or bidirectional, i.e., the plans are interdependent [Jen96, p. 192].

TÆMS TÆMS is an acronym for *Task Analysis, Environment Modeling, and Simulation* framework designed by Decker [Dec95, Chap. 3]. A task environment in TÆMS is described on three levels:

- the objective level,
- the subjective level, and
- the generative level.

On the objective level "the particular problem-solving situation or instance over time" [Dec95, p. 50] is modeled. That is, the overall situation is formally described, where the coordination has to be done. The tasks that have to be performed are modeled in a tree like form, i.e., they are decomposed into subtask. The leaves of this tree, the executable tasks, are called methods. Each method has a certain time interval, in which it has to be processed, a maximal quality that can be achieved, and a duration the execution requires to achieving maximal quality. Quality can only be achieved if the task is performed within its time interval. Note that method execution is preemptive, i.e., the execution of a method can be interrupted, gaining lower quality. The execution can be continued within time interval of the task. To model certain aspects of the environment it is possible to use so-called *local effects*. Those effects can be used to describe if a task can be interrupted and and how the gained quality is effected by the duration the task is executed. If a method is executed after its deadline no more quality can be achieved. Each (sub)task, that is not a method, has a quantifier that specifies how the quality gained from its subtasks has to be aggregated. Decker introduced four different functions to do so [Dec95, pp. 51.52]. For example taking the minimal quality of all sub-tasks can be used to model AND relationships. Taking the maximum quality of a sub-task can be used to model OR relationships. Other quantifiers introduced by Decker are the summation of qualities or the average

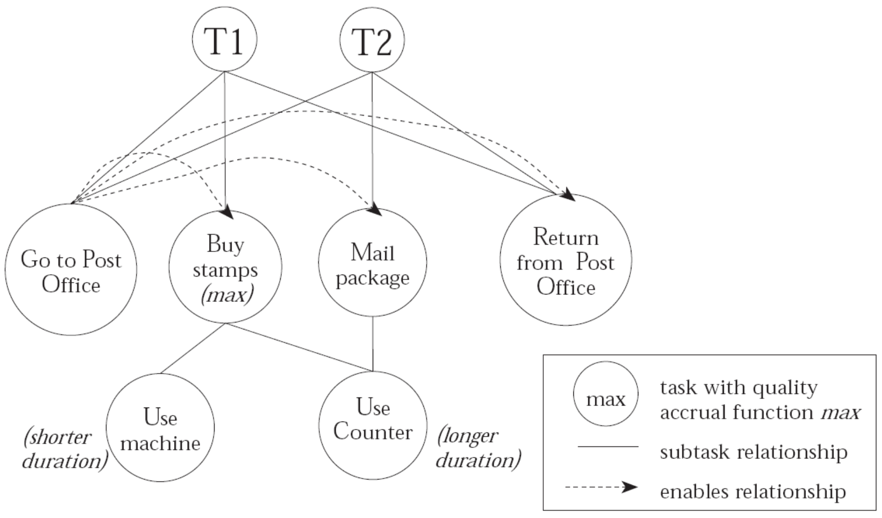


Figure 3.5.: Objective task structure in TÆMS, from [Dec95, p. 58]

quality of all subtasks. In the environment different tasks can exist simultaneously. Dependencies can exist between different task structures, as well. These dependencies can be modeled using *non-local effects*. Non-local effects are edges in the task graph, ending in the node that depends on the execution of the node the edge starts from. Non-local effects can effect duration or quality of a method. Decker [Dec95, pp. 55,56] propose the following set of non-local effects: *enables*, *facilitates*, *hinders*, *precedes*, *favors*, *cancels* and *causes*. Decker stated that these set can be extended if necessary. A resulting TÆMS objective structure is shown in Figure 3.5.

The subjective level of TÆMS is a mapping of the objective task structure to the part of the structure each agents can perceive. Thus, agents do not necessarily perceive the complete task structure or the correct task structure.

In the generative level of a TÆMS model characteristics of a certain domain, measured by statistical measures, are stored. The information on the generative levels could be used, to configure a generator of current instances of task groups.

A number of domains have been modeled in TÆMS. Among others, models of distributed sensor networks, hospital patient scheduling [Dec95, Chap. 3.6] and supply chain management [WGP03] have been presented.

Moreover, TÆMS has been extended into CTÆMS, mainly by adding a number of additional non-local effects. CTÆMS was used as modeling language in the recently finished DARPA project COORDINATOR[7]. In this project three research

[7]see `http://www.darpa.mil/ipto/programs/coor/coor.asp` Accessed: 02/11/2010

groups investigated coordination and planning techniques for various coordination tasks modeled in CTÆMS. Techniques that was used were hierarchical task networks by Smith et al. [SGZ+06] and Markov decision processes by Maheswaran et al. [MSB+08], for instance.

Task sharing is an idea origin from the research of cooperative distributed problem solving. A task of an agent can be decomposed into a number of sub-tasks. An agent can decompose a task into sub-tasks. These sub-tasks can be distributed among a number of agents that either perform the subtask or decompose and distribute them again. Thus, the main steps of this problem solving technique are, according to Durfee [Dur99, p. 124]:

- to decompose a complex task into simpler ones,
- allocate the subtasks to an appropriate agent,
- the subtask are accomplished in parallel, and
- finally, the results of the subtasks are combined and a result for the overall task is synthesized.

A well-known protocol that has been proposed for such situations is the contract net protocol [DS83]. Within the contract net protocol the contracting process is organized in form of a public announcement and a bidding process. One agent, that wants to share a task, takes the role of a *manager* that make a *task announcement*. If the manager has a prior knowledge about potential agents that can perform the task, he can send the task announcement only to these specific agents. If no such knowledge is available to the manger he can broadcast the announcement. Agents that receive an announcement evaluate if they can perform the task. If so, they send a bid to the manager specifying the conditions for performing the task. The manager might receive a number of bids. He can then select one bid according to a preference function. The successful bidder is informed via an award message.

If the manager does not receive a bid or no appropriate one, he can react according to Durfee [Dur99, pp. 128,129] in the following ways:

- broadcasting: If previously the manager has send the task announcement to a selected group of agents, he can broadcast the task announcement, to reach agents that he is not was aware of that they could perform the task as well.
- retry: The manager could, even periodically, retry to allocate the task to the agents, by periodically send out the same task announcement.

- announcement Revision: The manager might consider relaxing some constraints he has defined for the fulfillment of the task. Thus, if he relaxes constraints the task might become executable, or more attractive for other agents to perform.
- alternative decompositions: The manager can try to find another decomposition of the task that might be not optimal but resulting in a set of subtasks that can be distributed, to achieve the best implementable plan.

The contract net protocol is widely known and has also become part of the FIPA specification [FIP02f, FIP02g]. Moreover, the contract net protocol has been extended by Sandholm [San99] using marginal costs, which is explained in more detail in Section 3.1.3. In the following we discuss aspects of negotiations in multiagent systems.

Negotiations in MAS The notion of negotiation is often used in multiagent systems. Consequently, no coherent definition is available. We use here a definition given by Bussmann and Müller. According to their definition a negotiations is detailed as “the communication process of a group of agents in order to reach a mutually accepted agreement” [BM92]. Negotiations are typically used for task allocation or conflict resolution among agents [Mül96, p. 212]. According to Müller [Mül96, p. 213] research issuing negotiation can be grouped into the following classes:

- negotiation languages,
- negotiation decision, and
- negotiation process.

As pointed out in Section 2.3.4 negotiation between agents base on communication between the negotiating agents. Therefore, in earlier research concerning negotiations the topic of adequate negotiation languages has been addressed. Special languages, like COOL [BF95], have been developed for coordinating agents. As a consequence of the FIPA standardizing process most issues concerning negotiation languages, like languages primitives or languages for message exchange, have been standardized and are consequently no active research issue anymore. An exception are negotiation protocols, i.e., interaction protocols for negotiation. If FIPA interaction protocols[8] do not offer a natural fit or an efficient solution, additional more specialized protocols can be defined for negotiations.

[8]FIPA interaction protocols have been presented in Section 3.1.3 and summarized in Table 2.6 on page 52.

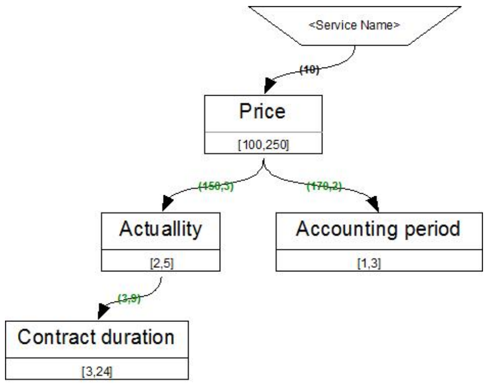


Figure 3.6.: Graphical representation of a trade-off relationship, see [SKT09] for more details

As any other communication process, negotiations have a content. The content of a negotiation is the issue or item the negotiation is about. This issue can either be described by one or more attributes that are regarded in the negotiation process. If only one attribute is the subject of the negotiation it is called a *single issue* negotiation otherwise it is called a *multiple issue* negotiation [Woo09, p. 316]. Single issue negotiations are easier to perform, as they boil down to the question which negotiation partner makes what concession. In multiple issue negotiation it becomes more complicated, as each negotiation partner has a set of private trade-off relations among the attributes that guide him in his negotiation process. A graphical representation of some trade-off relations, introduced in joint work with Kurtanovic and Timm [SKT09], is illustrated in Figure 3.6. Defining those trade-offs and then computing efficient offers to efficiently come to an agreement can become difficult, as pointed out in [SKT09]. In this article we provide a way to specify trade-offs in a human readable way that can automatically transformed into a representation that can be use by agents in negotiations. We have also presented techniques for efficiently computing bids. Therefore, we compute preference hierarchies offline and store them in a relational database. More details are provided in Schumann et al. [SKT09] and Kurtanovic [Kur08].

Negotiations can be sketched into three layers:

- The negotiation protocol: the protocol specify the possible sequences of messages, or more precisely, of performative.

- The negotiation behavior: the negotiation behavior specifies for each possible step in the negotiation process hot the agent should behave, i.e., what kind of message according to the protocol should be send next.

- Negotiation data: the data is on the one hand the data that is exchanged between the agents, and on the other hand also the data and functions that are required to provide enough information to the negotiation behavior.

A negotiation can end with a conflict or with a mutual agreement. These agreements are often referred to as commitments. In the following the role of commitments for the coordination among agents is discussed.

Commitments and conventions According to Jennings [Jen96, pp. 194,195] a *commitment* is a binding pledge an agent gives about its actions or beliefs. An agent can give a commitment to another agent, which reduce the uncertainty about the future behavior of the first agent, the second agent has. An example might be that agent A commit to agent B to perform task t in a certain time window. After this time windows has elapsed agent B assumes that the task t has been performed and eventually gets a confirmation by A. An agent can also give commitments to himself, which can be seen as an internal plan representation. Commitments can also be conditional, i.e., there exists a condition that has to be satisfied to activate the commitment. As aforementioned a commitment reduces the uncertainty about the behavior of an agent, but on the other hand the agent that has given the commitment has to reserve resources to perform the action he has committed to do and this limits its flexibility. Thus, an agent is limited in its future choices by its given commitment. If a number of agents wants to achieve a goal together that requires a joint action, i.e., an action that require a number of agents to perform it at the same time; these agents need a *joint commitment*. A joint commitment is a commitment that is given by all participating agents.

Even though commitments are binding it is possible for the agents to drop them. This can only be done with mutual agreement of all effected agents. A situation in which it makes sense to drop a commitment is if the commitment is satisfied or unreachable. Jennings [Jen96, p. 196] termed those rules to reason about commitments *conventions*. Additionally, Jennings [Jen96, p. 198] describes social conventions, which encode the rules for the agent how to behave towards other agents if its local reasoning about commitments has changed its opinion concerning a commitment. If the agent comes, by local reasoning, to the resolution that a joint commitment is not achievable anymore, it is not allowed to unilateral skip it by social conventions, for instance. A possible social convention might be that the agent has to inform all other participating agent of the joint commitment that the current joint action will not be successful, according to its local reasoning.

According to Jennings [Jen96, p. 200] commitments and conventions are essential for the coordination among agents. Therefore, Jennings formulated the following equation [Jen96, p. 200]:

$$\text{Coordination} = \text{commitments} + \text{conventions} + \text{social conventions} + \text{local reasoning}.$$

Jennings encoded all conventions in forms of rules. This enables him to build agents using rule-based systems to coordinate the behavior of the agents. These ideas were implemented in the ARCHON system, a short overview can be found in Wooldridge [Woo09, p. 168] and a more elaborated description is provided by Cockburn and Jennings [CJ96]. Another interesting example has been presented by Busetta et al. [BMRL03].

Result sharing By task sharing, a set of task can be allocated to a number of agents. Result sharing becomes necessary if the different task depend on each other or influence each other. Result sharing enables agents to achieve better performance. Durfee [Dur99, p. 131] has characterized four kinds of positive effects that could be achieved:

- Confidence: if multiple agents have computed the same results their confidence in their results can increase.
- Completeness: if the agents have computed results for subtask they can gain a more complete plan by combining their solutions.
- Precision: if agents consider actions taken by other agents into account in their planning process, they can generate more coherent plans.
- Timeliness: if results from other agents restrict the search space of an agent, it can generate its results faster.

There exist different strategies for the communication of the results to different agents, ranging from broadcasting, which is typically very costly in terms of communication costs, to single point to point communication. So in a contract net like task/subtask topology, results are typically reported back to the manager that contracted the current task. Other approaches use the idea of building structuring organizational structure in MAS to guide the flow of information and existing dependencies. Result sharing can either done during the planning process, i.e., partial plans are shared; or after the local planning systems have finished their computation. Examples of result sharing during the planning process are forwarding techniques like *forward checking* in (distributed) constraint satisfaction, see Yokoo [Yok01, p. 11]. A more detailed discussion of result sharing is provided by Durfee [Dur99].

For the coordination of planning entities, and thus plans, result sharing during the planning process is not well suited. As already mentioned, in Section 2.2.3

planning systems can show chaotic behavior. Thus, sharing plans in progress cannot be used to achieve coordination among planning systems, as the published plan can still drastically change during the local planning process.

The idea of first completing local planning and then share the results, and if necessary, adapt the plans is applied, for instance, in von Martials work addressing the coordination of plans from different agents [Mar92, Chap. 3].

Generalized Partial Global Planning (GPGP) Decker [Dec95, Chap. 5] introduced GPGP as an extension of the partial global planning (PGP) from Durfee[9]. GPGP was designed to work in environments described in TÆMS. A driving idea of GPGP was to separate the coordination procedure from the local scheduling system [Dec95, p. 100]. GPGP comprises a set of coordination mechanisms for a team of cooperative computational agents. According to Decker [Dec95, p. 117] the coordination mechanism has to be detailed concerning aspects like:

- how and when communicate/construct non-local views,
- how and when exchange partial results, and
- how and when make and break commitments.

A non-local view is an enriched representation of the objective environment an agent can build using the subjective view of the agent plus additional information received from other agents. GPGP emphasizes result sharing and information sharing about the task structure, thus meta-information, as well. Therefore, it is not a strict result sharing approach. For applying the GPGP mechanism it has to be configured. Configuration is done via specifying strategies for five different coordination activities. These activities are [Dec95, Chap. 5.6]:

- Updating non-local viewpoints: Defines what degree of local information about the problem state has to be published, e.g., all, nothing or fractional.
- Communicating results: Defines the strategy that is used to send results to other agents, e.g., minimal to satisfy commitments, all results or send results to interested agents.
- Handling simple redundancy: Defines the strategy how to deal with duplicated redundancy in agent's plan, e.g., a redundant task is performed by one arbitrary chosen agent out off the agents that wants to perform the task, and results are communicated to all of them.

[9]An introduction to PGP is provided by Durfee in [Dur99, pp. 153–157].

- Handling hard coordination relationships: Define the strategy how to deal with the violation of hard coordination relationships: e.g., perform a local replanning.

- Handling soft coordination relationships: Define the strategy how to deal with the violation of soft coordination relationships: e.g., check if a replanning is reasonable or not.

Note that hard and soft coordination relationships have been discussed above. As a consequence of the flexibility GPGP offers in configure, the coordination algorithm can be detailed to a *family of coordination mechanisms*. According to Decker [Dec95, p. 125] the GPGP mechanism can be configured in 72 ways, leading to different results for a given environment at hand. Thus, the mechanism can be specialized for its target environment.

GPGP has two kinds of commitments, local and non-local ones. Thereby, non-local commitments are given to other agents. A local schedule is generated by the local scheduling system that has to try to regard all commitments, locals and non-locals. If the plan is computed, the coordination mechanism identifies which non-local commitments could not be satisfied, these commitments have to be resolved, as they cannot be fulfilled anymore. While local commitments that could be satisfied can be published to other agents, and thus become non-local commitments [Dec95, p. 127].

GPGP, as well as TÆMS, have been applied in a number of domains, like distributed vehicle monitoring by Decker [Dec95], hospital patient scheduling by Decker and Li [DL00] and supply chain management by Wagner et al. [WGP03].

Coordination as a planning problem Another perspective to tackle coordination problems is to see them as a planning problem. Durfee distinguished three different cases [Dur99, Chap. 3.5]. These cases are presented in the following.

Centralized planning for decentralized execution The main idea is that there exists a centralized planner that has all required knowledge to compute a global plan. This kind of approach has communalities with the top-down and top-down/bottom-up planning approach presented in Section 3.1.1. Consequently, the same problems arise, for instance, the centralized planner requires a large amounts of information to perform its task. This information has to be kept updated over time, which is another problem and the centralized planner has to have the power to enforce that the agents commit to its plans. Due to these problems this type of coordination is not focused in this study.

Decentralized planning form a global plan The idea of this approach is that a number of *experts* compute together a plan. Durfee [Dur99, p. 141] especially mention here planning problems from the fields of supply chain and logistics. Nevertheless, in the example he presents the planning was performed in a sequential way, i.e., results were passed from one planning agent to another. Limitations of this sequential solution generation have already been discussed in Section 3.1.1. If there exist interdependencies, and thus cyclic dependencies in the dependency graph, this approach cannot be applied. As shown, for instance, by Meier and Schumann [MS07], cyclic dependencies can be found in real world planning and coordination problems, and are not purely of academic interest. Thus this approach might be used solving problems with linear dependencies, only. But it is not sufficient for problem solving, when the coordination problem has cyclic dependencies.

Decentralized planning for decentralized execution This approach addresses situations in which the plans are generated by a number of planning systems and are executed also in a distributed manner. Thus, it might be even not necessary to form a global plan. Nevertheless, avoiding unwanted plan interactions raises the need for coordination among the plans. There exist a number of approaches for doing so:

- **Pre-planning coordination**: To avoid conflicting results an idea might be to eliminate the possibility of interference of partial plans at all. The interference of plans are a consequence of the dependencies that exist between the tasks/actions that have to be performed. Witteveen and a number of researchers from his group have taken up this approach. The main idea is to add a number of constraints into the original problem formulation to avoid interfering plans. The existing potentials for parallel execution should be exploited and possible interference (caused by a too high degree of distribution) can be avoided. Unfortunately, the problem of adding additional constraints is a very hard one, for details see Steenhuisen and Witteveen [SW07] and Witteveen et al. [WHR09].)9

- **Plan merging**: The idea of plan merging origins from classical AI planning, especially STRIPS planning (see Section 2.2.1). Each action has additional information about what conditions have to be satisfied while the action is executed. For each intersection between two plans, it has to be checked if the different plans interfere with each other. The intersection of two plans are all their possible interference points. If a negative interference is detected they have to be removed, either by synchronizing the plans, or revising them, if synchronization is not possible. An introduction into plan merging can be found in Durfee [Dur99, p. 141] and more elaborated work has been presented by de Weerdt [Wee03].

Plan merging requires a central entity that is capable to reason about the interdependencies of the plans and means to resolve those interdependencies. This is hard to achieve for domains where the necessary knowledge cannot be formalized easily.

- **Mediator-based approaches**: Another way, also using a central entity is based on the idea of using a mediator. A mediator can observe the overall systems behavior and can temporally limit the autonomy of the agents and thus order them to perform certain policies or it can solve subproblems in a centralized manner more efficiently. Examples for such mediated approaches have been presented by Schumann, Lattner and Timm in the research concerning *regulated autonomy* [SLT08c, SLT08b]. Another approach for mediation in the field of distributed constraint satisfaction (see below) has been presented by Mailer [Mai04].

- **Iterative plan formation**: The idea of this approach is based on the assumption that conflicts between agents have to be removed. By iteratively remove conflicts the plans become more and more coherent. A detailed description of these techniques is provided by Durfee [Dur99, p. 144–148]. The idea of iteratively generate plans that leads to more and more coherent plans is appealing. For plans that depend on each other, the output of one planning system determines the inputs for another planner, a corresponding method for iterative plan formation would be a fix point iteration. This idea has been investigated by Meier and Schumann in [MS07]. It could be shown, as it was expected, that due to the chaotic behavior of planning systems[10], no such fix point could be computed and that no narrowing pattern could be identified after multiple iterations.

- **Negotiations**: As already mentioned, negotiations are used for conflict resolution as well. They are a common technique to harmonize distributed generated plans. Negotiations in general have already been discussed above. So here two applications of negotiation designed in the context of the German priority research program “Intelligent Agents and Realistic Commercial Application Scenarios” [KHLS06] are presented. Paulussen et al. [PJDH03] present a negotiation scheme in the domain of hospital patient scheduling. Agents representing patients negotiate with resource agents representing hospital units which offers services. An artificial currency is introduced to model

[10]see Section 2.2.3

a single-item negotiation among rational agents. An approach in the domain of supply chain management is presented by Frey et al. [FSWZ03]. In their approach they coordinate different agent-based planning systems. Each of those planning systems hosts a coordination agent that can inspect the local plan and use these data to negotiate efficiently.

Coordination as distributed constraint satisfaction Problem A set of tasks and dependencies can also be formulated as a distributed constraint satisfaction problem (DCSP) [LS96]. A constraint satisfaction problem [GNT04, p. 168] is defined by a finite set of variables $X = \{x_1, ..., x_n\}$; a finite set of domains $D = \{D_1, ..., D_n\}$ for the variables with $x_i \in D_i$; and a finite set of constraints $C = \{c_1, ..., c_m\}$. Each constraint c_i involves a subset of variables and specifies allowed combination of values for that subset of variables. A constraint is defined by a predicate [Yok01, p. 2].

A solution for a (distributed) constraint satisfaction problem is an assignment $v = \{v_1, ..., v_n\}$ to the variables X such that $v_i \in D_i$ and all constraints are satisfied.

In a DCSP the variables are distributed among different agents. A constraint c_i is known to all agents that are assigned to variables that are effected by the constraint.

Liu and Sycara [LS96] model the coordination problem in a job shop scheduling sequence (see Section 2.2.1) in form of a DCSP. They do so by defining a variable for the start time of each operation that has to be performed. The precedence relations between operations are encoded in the constraint set as well as the capacity restrictions. Another approach for the coordination of distributed schedules has been presented by Kawamura et al. [KKAO00]. In their approach they try to find a solution via negotiation among agents representing different planning systems. These agents have to ensure that local constraints (e.g., capacity constraints) and global constraints (e.g., sequence of activities) are satisfied.

The field of efficiently solving those kinds of problems is not in the focus of this research and thus not explained here. A detailed introduction into DCSP and algorithms to solve those problems is given by Yokoo [Yok01].

Coordination from the perspective of distributed computing

Coordination was investigated in the field of distributed computing, too. This research has also influenced the AI research. Here, a short overview about research in distributed systems is given and shown how this interrelated to multiagent systems research. A work that tried to bridge both research fields was presented by Schumacher [Sch01].

Coordination in distributed computing is motivated by the need of data exchange of threads/processes among each other and their synchronization. According to Schumacher [Sch01, p. 15] the coordination approaches, presented so far, suffering from their agent-centric perspective. Therefore, Schumacher termed these approaches subjective coordination. He argues that the environment of the agents, as the coordination media, has to be respected at design time as well. In contrast to the subjective coordination approaches Schumacher termed the coordination with a more system-wide perspective the objective coordination [Sch01, p. 17].

Coordination from the perspective of distributed computing can be described in form of a *coordination model*. A coordination model comprises [Sch01, p. 34]:

- the *coordinable entities*, which are the concurrently running activities,
- a *coordination medium*, which enables the coordination and can be described as the place where the coordination took place, and
- the *coordination laws*, which specify the semantics of the model.

Such models can be expressed in form of coordination languages. Thereby, coordination languages are defined as follows: "A coordination language is the materialization of the linguistic embodiment of a coordination model" [Sch01, p. 35]. Coordination models can be either data-driven, process-oriented or hybrid. A well-known data-driven model is LINDA. An introduction into LINDA can be found, e.g., in [Sch01, Chap 3.3]. In the LINDA model processes communicate via a shared data space. The items that are communicated are tuples, whereby a tuple can contain a number of data entries each a simple data type. These tuples are referred to as passive tuples. A special type of tuple is an active tuple. An active tuple is a process in the tuple space that computes a result, if it terminates the result becomes available as a passive tuple in the tuple space. Information is retrieved from the tuple space using a template matching mechanism. In such a template fields can either by filled with a constant value or with a placeholder for a value. Information is retrieved from tuples in the tuple space that satisfy the characteristics described in the template. Linda defines a number of primitives for interacting with the tuple space like read, write or create active tuples.

LINDA had major influence on the field of coordination in distributed computing and its ideas, among others, have taken up into mainstream programming languages, like the *Java Spaces* that have been developed in the efforts developing *Jini*[11].

The second class of coordination modes are process-oriented models. They describe the state of the computation. These models enables to distinct between

[11]see `http://java.sun.com/docs/books/jini/javaspaces/`, Accessed: 02/13/2010

the computation and the required coordination. Different types of channels can be defined and thus different types of interaction among processes, like blocking communication and non-blocking. A well-known example for a process-oriented coordination language is Manifold, described, e.g., by Schumacher [Sch01, pp. 42,43].

In his work Schumacher [Sch01] present the ECM coordination model, that can serves as a meta model for a number of concrete coordination languages. One of these languages is the SLT++ language. In the SLT++ languages the communication is realized via connections that can connect an input and an output port of a process/agent. An agent that sends a message on a channel does not know which agents will listen to this channel, or if an listener exists at all. Note that consequently all communication is anonym, which contrast the typical assumptions in multiagent communication presented in Section 2.3.4.

Objective and subjective coordination

The idea of objective coordination has influenced research of coordination in DAI and lead to a number of results especially dealing with the ideas of an agent environment that could facilitate coordination between agents.

Combination of objective and subjective coordination approaches A similar concepts to the aforementioned distinction between objective and subjective coordination is the distinction between the *micro-level* (agent-centric) and the macro-level (MAS-centric) that is favored by Omicini and Ossowski [OO03]. Both concepts have been compared, e.g., by Bergenti and Ricci [BR02] and by Omicini and Ossowski in [OO03], for instance. In the following research approaches to integrate those two perspectives that have been proposed, e.g., by Omicini et al. [ORVR04].

As pointed out by Ricci et al. [RVO06] not all entities in a system have to be represented as agents. Therefore, Ricci et al. propose the use of *artifacts* as an additional concept for modeling multiagent systems. Artifacts are part of the environment of agents and can be used by them. A special type of these artifacts are *coordination artifacts* , discussed, e.g., by Omicini et al. and Ricci et al. [ORV$^+$04, RVO06]. A coordination artifact corresponds to a coordination medium, mentioned above. A coordination artifact is an element of the agents environment. Agent can use the artifacts to coordinate their behavior. Thus, these artifacts encapsulate a *coordination service* that can be reused and can be a building block during the design of a MAS. Consequently, it is argued by Ricci et al. [RVO06] that coordination artifacts are an engineered approach to coordination. A coordination artifact, like any coordination media, should exhibit the following properties [ORV$^+$04]:

- Specialization: agent should coordinate their activities effectively and efficiently.
- Encapsulation: the coordination mechanism is encapsulated and thus can be reused.
- Malleability: the coordination behavior can be adapted at runtime.
- Inspectability: the behavior should be inspectable and observable to enable maintenance and debugging.
- Predictability: the coordination should be performed predictable, i.e., provable, exactly confirming to the coordination laws.

It goes without saying, that these properties are typically not satisfied by intelligent agents (see Section 2.3.1). Thus, these artifacts are objects in the multiagent environment. The idea to enriching the agents environment with artifacts is used in recent research creating embodied organizations, described, e.g., by Piunit et al. [PRBH09], too.

An extension of this idea has been presented by Vilencia et al. [VSL$^+$10]. Thereby, an agent framework should be enriched by an extra coordination layer [SR08], that handles the coordination explicitly. Thereby, coordination spaces similar to coordination artifacts are used within this layer. Embedding these coordination spaces in the environment should free the agents from reasoning which coordination artifact they should use [VSL$^+$10]. Therefore, coordination spaces have listeners that can intercepts events that occur inside the agent. Results of the coordination can be send via events, to the agent. The agents perceive those events as an perception from their environment.

The scope of application for coordination artifacts has been described by Ricci et al. as follows: "coordination artifacts ... basically works when it is possible and useful to design a priori the solution to the coordination problem" [RVO06, p. 211]. Examples of coordination mechanisms embedded in coordination artifacts discussed in the literature are for instance, the follow-me artifact, where on agent imitates another one, presented by Omicini et al. [ORV$^+$04]; or synchronization demonstrated by the dining philosophers problem used by Ricci et al. [RVO06]. Moreover pheromones, used in, e.g., in ant algorithms, have been represented as coordination artifacts/spaces proposed by Vilencia et al. [VSL$^+$10].

Omicini et al. [OOR04, pp. 284–286] points out that the integration of objective and subjective coordination has been inspired by the activity theory. Activity theory origins from social psychology. According to the activity theory three different layer of cooperative activities can be identified:

- Co-ordinated: thereby participants follow their fixed roles aiming to successfully performing the activity.
- Co-operative: actors focussing on a common objective, thereby the roles of the participants are not fixed. The actors have to balance their activities regarding possible interdependencies.
- Co-constructive: actors focus on re-conceptualisation of their organization and their interaction, like the roles used in co-ordinated actions.

In the transfer to MAS the authors argue that co-constructive actions are done by agents that reason about the objectives of the MAS and thus define social objectives and task that have to be performed. In co-operative activities agents design coordination artifacts, either embodied (coordination media) or disembodied (plans, interaction protocols), in order to achieve the tasks and coordinate their action. Consequently, within the co-ordinated activities agents can use the coordination artifacts to coordinate their behavior.

Omicini et al. [OOR04] advocate that in contrast to existing agent platforms should offer a richer feature set. Currently agent platforms enables agents based on these platforms to communicate. Additional features Omicini et al. propose are techniques for coordination as build-in features of the platform. This would offer a higher level of abstraction for building multiagent systems. It would also increase the reuse of these basic coordination mechanisms, as they are part of the platform the agents are hosted in. The authors [OOR04, Chap. 6] mention the TuCSoN platform[12] as an example for such an extended platform. TuCSoN provides coordination services that base on a tuple centered coordination model, which has been outlined above.

Those models enable agents to reason about an appropriate coordination medium/artifact to coordinate their behavior among each other. An example how such reasoning can be performed has been presented by Excelente-Toledo and Jennings [ETJ04].

Recent joint research effort Recently, the research concerning coordination in multiagent systems has been facilitated by a joint international research effort. This is in the COST Action ICO801 “Agreement Technologies”[13]. The COST action has been established in 2008 and will continue to 2012. In this actions five work groups have been defined that concentrate on current research topics. The issues are:

- semantics,

[12]for details see `http://apice.unibo.it/xwiki/bin/view/TuCSoN/`, Accessed: 02/13/2010
[13]`http://www.agreement-technologies.eu/`, accessible at 02/13/2010

- norms,
- organization,
- argumentation/negotiation, and
- trust.

In the following the goals of these working groups are presented. A more detailed presentation of this project is given by Ossowski [Oss08].

Semantics As a prerequisite for agreements is successfull communication. Therefore, it is on the one hand necessary to transfer messages correctly, and on the other hand enables correct interpretation of the content of the message. Consequently, semantics is a central research topic enabling to come to agreements. To achieve mutual understanding between agents techniques like ontology alignment and the detection and solution of semantic mismatches have to be addressed.

Norms Norms, as well as social laws, regulates agent's behavior. Note that agent can break norms if it is rational for them to do so. These norms can be either given by the systems designer or can emerge and evolve from the system's behavior [Woo09, p. 173]. Both aspects require systematic approaches to achieve desired behavior. An approach for norm evolution has been proposed, e.g., by Timm, Lattner and Schumann [TLS09].

Organization As already mentioned, in Section 3.1.1 an organizational structuring of agents can reduce the coordination complexity. Consequently, the use of organizations as a technique to coordinate agent's behavior has to be explored as well. An organization in a MAS is composed out of a set of roles that agents can fulfill and a set of rules, often referred to as social laws or norms, that regulate the interaction among roles [Jen96, p. 204]. The design of organizations has been subject of research for a long time (see [CG99]). Research concerning organization in MAS address on the one hand the formation of organization, like team formation or coalition formation, and on the other hand the evolution of organizational structures. Beside the pure methodologies tools support is necessary as well, to ensure that these concepts and principles become useful in practical applications. Thus, it is essential that tool support is given.

Argumentation and Negotiation A key theme in negotiations is the exchange of agent's positions and making concessions to each other. As already mentioned, this process is termed negotiation. Within an argumentation an agent tries to convince its opponent to change its position [PSJ98]. This is typically realized using logics, i.e., agents try to manipulate the belief set[14] of the agent, so that the position of the other agent changes. Argumentation is already an issue for research. An integration of argumentation techniques into negotiation approaches seem reasonable to enrich negotiation techniques. This integration has to respect the fact that for practical applications these negotiation techniques have to be realized regarding limited resources and limited time.

Trust As trust is a prerequisite for finding agreements between actors, it is an issue for the research concerning agreement/coordination. Trust between agents establishes by positive experience with another agent, i.e., the other agent behave as expected. To establish trust agents have to identify similar situations, to decide if they can trust another agent or not in a new situation. While norms have a rather static nature, they evolve over time slowly; trust can be used to offer more flexibility for the agents. Consequently, norms and trust mechanisms have to be balanced. Additionally, semantics support for trust is required.

3.1.3. Coordination concepts in game theory and mechanism design

Research concerning coordination, especially in DAI, has been heavily influenced by game theory. Papers concerning game theoretical issues, like mechanism design, can be found with a significant proportion in major DAI conferences, like the AAMAS[15].

According to Holler and Illing [HI06, p. 1] the focus of game theory is the analysis of strategy decision making. Thereby, it is assumed that different autonomous rational decision takers participate in the decision making process and the final outcome depends on the interdependence between the different decision chosen by the participants. Thereby, conflicts of interests or coordination problems can occur. In game theory these decision problems are modeled as games. Beside the rationality assumption it is typically assumed that all players have full information about the possible outcomes of a game, even though these assumptions can be partially relaxed, e.g., by applying mixed strategies. The field of game theory can be distinct into game theory and mechanism design. In the subfield of game theory the strategy rational agents apply in a game is in the focus of investigation. In the field of mechanism design the perspective is different. Given the assumption of rational agents, the question is, how mechanisms/games should be designed to

[14]Belief sets are discussed in Section 3.2.1.

[15]See, for instance, for program of AAMAS 2008 (`http://gaips.inesc-id.pt/aamas2008/tech_programme.html` Accessed: 02-09-2010) or the list of accepted papers of AAMAS 2009 (`http://www.conferences.hu/AAMAS2009/accepted_papers.html` Accessed: 02-09-2010)

guarantee certain properties. Both branches are presented in the following.

A rather short introductions into game theory in AI and in particular in multiagent systems is given by Russell and Norvig [RN03, Chap. 17.6], Sandholm [San99] and Wooldridge [Woo09, Chap. 11-15]. A more detailed introduction from the perspective of DAI into game theory is provided by Shoham and Leyton-Brown [SLB09]. Another elaborated introduction from an algorithmic perspective has been edited by Nisan et al. [NRTV07].

Game theory

The field of game theory can be further subdivided into the research concerning non-cooperative and cooperative games .

In non-cooperative games a detailed investigation about what actions are chosen in each round of the game is done. It is not possible for the agents to commit themselves to a strategy before the game starts. Thus, the agents cannot make deals and form coalitions. The goal of non-cooperative games is to investigate the best possible, i.e., utility maximizing, strategies for the players.

Cooperative games have a more abstract perspective on games. The actions of each players are not model in detail. Instead the agent knew only its payoff. Agents are allowed to commit to a strategy and therefore can form coalitions to maximize their expected profit. Consequently, cooperative games lay the fundament for a formal approach to investigate coalition forming among self interested, rational agents. In the focus of this study is the coordination of agents and not the process how they come to a coalition. A game in game theory is given by:

- a set of players,
- a set actions the players can perform, and
- a payoff matrix, that gives the utility for each player for each combination of actions.

The probably most widely known game is the *prisoner's dilemma*. Two criminals A and B have been caught by the police. They are interrogated separately and have no means of communication. They can either confess or stay silent during their interrogation. If both confess the crime they will be arrested for 5 years, each. If both stay silent the can only be arrested for 1 year, each. If only one confesses the other will be send to jail for 10 years while the other one goes free. The payoff matrix of this game is shown in Table 3.2

In the analysis of games a number of criteria are focused that are introduced here briefly, a more elaborated discussion of these criterions is presented by Sandholm [San99].

	A confess	**A stay silent**
B confess	$A = -5$, $B = -5$	$A = -10$, $B = 0$
B stay silent	$A = 0$, $B = -10$	$A = -1$, $B = -1$

Table 3.2.: Payoff matrix of the prisoner's dilemma, [RN03, p. 633]

- Social welfare: Is defined as the sum of all agent's payoffs/ utility in a given solution.
- Pareto efficiency: A solution x is Pareto efficient if there exists no solution x' such that at least one agent can do better and no other agent is worse off.
- Individual rationality: The agents payoff is higher when he participate in the negotiation, than if he would not participate.
- stability: The agents should not be rewarded for manipulating a game.
- Computational efficiency: The computational effort should not be high when participating in a game.
- Distribution and communication efficiency: Games should not have a single point of failure or require a lot of communication effort, typically measured in number of messages.

Each player has to choose a *strategy*. A strategy is a complete definition how the player wants to play its game, i.e., it specifies which action has to be taken at what situation. A strategy is called a *pure strategy*, if the action selection is not influenced by random; otherwise the strategy is called *mixed strategy*. A strategy profile ($s = (s_1, \dots s_n)$) is an assignment that assigns a strategy to each player. A strategy is rational for an agent, if it is rational for the agent to apply the strategy for its decision making. Thus the strategy maximize the utility of the agent. A strategy profile is a *solution* if for each player the assigned strategy is rational.

A strategy s_i of a player i is called *dominant strategy* if whatever strategy s_j an other agent j choose, the outcome of i will be at least as good as if he would have played another strategy. Thus, the strategy s_i specifies the best response to all possible strategies of agent j. The dominant strategy for both players in the aforementioned prisoner's dilemma is to confess. It is dominant for both players because the payoff matrix is symmetric. Which will lead in the outcome $A = -5$, $B = -5$ which is obvious not a solution maximizing the social welfare, and is not Pareto optimal/efficient.

Another concept leading to stable strategy is a *Nash equilibria*. A strategy profile ($s = (s_1, \dots s_n)$) is a Nash equilibria if for all agents i, s_i is the best

	i head	**i tail**
j head	$i = 1, j = -1$	$i = -1, j = 1$
j tail	$i = -1, j = 1$	$i = 1, j = -1$

Table 3.3.: Payoff matrix of the penny matching game, [SLB09, p. 58]

	left	**right**
left	1,1	0,0
right	0,0	1,1

Table 3.4.: Payoff matrix of the road side choosing game, [SLB09, p. 57]

response to all other strategies in s [SLB09, p. 62]. Obviously, the dominate strategy of the prisoner's dilemma is a Nash Equilibrium, because each agent can only get worse by unilateral deviating from the dominant strategy. But the prisoner's dilemma also shows another property of the Nash equilibrium, it is not unique. The strategy where both prisoners stay silent is a Nash equilibrium, too. In fact the outcome $A = -1$, $B = -1$ would be better, i.e., the Pareto efficient solution, maximizing the social welfare. It is possible that no equilibrium exists, at all, for instance, for the penny matching game[16]. Two players i and j choose simultaneously the face of a coin, either head or tail. If they had chosen the same side player i wins, otherwise j wins the game. The resulting payoff matrix is shown in Table 3.3. The observation that a number of games exists, where no unique Nash equilibrium exists motivates the class of coordination games, also called common-payoff games.

Coordination games A coordination problem arises when a game has multiple equilibria [MSW02, p. 399]. In such games the agents have no conflicting interests; their sole challenge is to coordinate on an action that is maximally beneficial to all [SLB09, pp. 56,57]. An example of a coordination game is the decision on which side of a road, drivers should drive [SLB09, p. 57]. They can either choose both the right or left lain and no crashes will occur. As opposed to if both decide to choose divergent side of the road and will crash if they come ahead on a road. The payoff matrix of such a game is shown in Table 3.4. A well-known coordination game is the *Battle of Sexes* (see [SLB09, pp. 58–59]). A couple has to decide which movie they want to see at the cinema. While he prefers an action movie, she prefers a romantic movie. They both do not want to go to one of the films

[16]The penny matching game is taken from Wooldridge [Woo09, p. 231]. It is equivalent to the two finger Morra game, presented by Russell and Norvig [Woo09, p. 632].

	action movie	**romantic movie**
action movie	2,1	0,0
romantic movie	0,0	1,2

Table 3.5.: Payoff matrix of the battle of sexes, [SLB09, p. 59]

	M: DVD	**M: CD**
P: DVD	$M = 9$, $P = 9$	$M = -4$, $P = -1$
P: CD	$M = -3$, $P = -1$	$M = 5$, $P = 5$

Table 3.6.: Payoff matrix of the video technology decision, [RN03, p. 634]

alone. The payoff matrix of this game can look like shown in Table 3.5. Beside the two pure Nash equilibria there exist also mixed Nash equilibria. That is, the actions are chosen with a probability. This, of course, implies that the game is repeated, theoretically over an infinite number of iterations. In such cases the application of mixed strategies can solve the coordination game, in a rather *fair* way for both players.

But there might be games that cannot be repeated over and over again. For instance, decisions concerning technical development like standards. Russell and Norvig [RN03, pp. 634–635] detail an example where a video game hardware manufacturer (M) has to decide if its next machine will be equipped with a CD or DVD drive. While the video game software provider (P) has to decide on which medium it next game will be released. The resulting payoff matrix is given in Table 3.6. Note that only one of the both existing Nash equilibria is Pareto efficient.

Other example of coordination problems that have been investigated by game theory are discussed in [MSW02, p. 401]. They mention different studies where different standards competing against each other have been investigated, for example the QWERTY keyboards or the VHS vs. Betamax formats for video tapes. In both coordination scenarios two different standards exists and people have to coordinate themselves to a single standard.

In contrast to the coordination games there exist also the so called *anti-coordination games* [SLB09, p. 198]. Where each player has to choose another option than the other did. An example of such a game is routing in networks, where two packages should be routed different way, instead of interfering. The following Table 3.7 illustrates the payoff matrix of an anti-coordination game.

More elaborated discussions concerning equilibria of games and coordination games can be found in Shoham and Leyton-Brown [SLB09] and Nisan et al. [NRTV07].

	left	**right**
left	0,0	1,1
right	1,1	0,0

Table 3.7.: Payoff matrix of an anti-coordination game, [SLB09, p. 198]

Mechanism design

The subject of mechanism design focuses on the design and analysis of economic mechanisms. Such a mechanism is called a protocol. But it has another semantic in contrast to the interaction protocols mentioned earlier in Section 2.3.4. A protocol in game theoretical sense is not a description of the syntactical sequence of messages, here offers. It raises requirements concerning the subject of the offers. These mechanisms can be used to design coordination mechanisms for a number of problems, for instance, allocation problems, among self-interested agents. Consequently, the field of mechanism design has influenced research in DAI, significantly. Another reason might be that due to its engineering approach and its solid mathematical foundation it might be appealing for computer scientist to use. Prominent types of mechanisms that are typically investigated in mechanism design are:

- voting,
- markets,
- auctions, and
- government policy.

As this thesis is concerned with the coordination of autonomous planning entities, we address here different types of auctions and markets. For a general introduction into mechanism design we refer to Nisan et al. [NRTV07, Part II].

Auctions Auctions are a well-known economic mechanism in computer science. Auctions are typically easy to explain, i.e., the protocols are not complex, and are distribution and communication efficient. Auctions are typically used to allocate scare resources. In such cases it is desired that the resource is allocated in the most efficient way. That is, the resource is allocated to the bidder that evaluates this resource most, i.e., typically it is assumed, that this bidder can gain the highest profit with the resource, or in case of task distribution can handle this task most efficiently. A key idea of auctions is that the object being auctioned has a value. This value can be either a common value, i.e., it is known and equal for all bidders, or private, i.e., it can differ for each bidder. Of course, it can happen that the evaluation of the good of an agent depends also on external factors, like the reselling price it can achieve for the good. The resulting value is then called correlated. Based on these values a price is fixed the successful bidder has to pay. Auctions form a sub-class of structured negotiations [SLB09, p. 317].

Auctions can be divided into single-item auctions and multi-item auctions. These types vary drastically in their complexity. Single-item auctions can be suitable for simpler coordination problems. Well-known types of single-item auctions are the English auction, the Dutch Auction, or the Vickrey auction, for instance. A detailed introduction into the field of auctions, and in particular single-item auctions, from the DAI perspective is given by Sandholm [San99, pp. 211–219], Wooldridge [Woo09, Chap. 14], Shoham and Leyton-Brown [SLB09, Chap. 11], and Nisan et al. [NRTV07].

The single-item auctions are not appropriate if the items that are auctions can exhibit substitutability or complementarity effects in the valuation of a bidder.

Substitutability can be formalized as follows. Let G be the set of all goods and v the evaluation function of a bidder. Then the subsets $G_1, G_2 \subseteq G$ with $G_1 \cap G_2 \neq \emptyset$ then G_1 and G_2 are substitutability iff

$$v(G_1 \cup G_2) < v(G_1) + v(G_2).$$

In contrast the opposite relationship, when the sum of its subparts is higher evaluated than the overall set of goods is called complementarity. Let us assume that the subsets $G_1, G_2 \subseteq G$ with $G_1 \cap G_2 \neq \emptyset$ then G_1 and G_2 are complementarity iff

$$v(G_1 \cup G_2) > v(G_1) + v(G_2).$$

If such relationships exist between the items to be allocated single-item auctions are not sufficient. They would lead to inefficient allocations. A well known approach to tackle such situations are combinatorial auctions. The main idea of combinatorial auctions is that the goods are grouped into bundles. A bundle is a non-empty element of the powerset of the set of all goods. Each bidder has to bid for each bundle. For example, if the task is to assign three different goods 1, 2 and 3 then there exist seven different bundles. If there exist three different bidders (A,B,C) for these goods the resulting matrix of bids contains 21 entries. A corresponding example is shown in Table 3.8, which is taken from Ruß and Vierke [RV99]. In this type of auction an issue arises, that has not been a serious problem in single-item auctions, the *winner determination problem*. For the auction presented in Table 3.8 the winning bids are shown in bold letters. In the auction presented so far, the winner could be determined simply by sorting the bids according to their price. This could be done rather easily with $n\ log(n)$ time

Agent	Bundles						
	{1}	{2}	{3}	{1,2}	{1,3}	{2,3}	{1,2,3}
A	80	60	50	90	110	100	150
B	30	90	20	120	**50**	100	160
C	80	**40**	40	110	80	90	180

Table 3.8.: Bid matrix for a combinatorial auction [RV99]

complexity, where n corresponds to the number of bids. In a combinatorial auction more than one winner can exist. The winners of a combinatorial auction are identified by a set of bids B that must fulfill the following conditions. Each good g ($g \in G$) must be contained by exactly one bid $b \in B$. Thus, each good is assigned. Second, only one bid can be assigned to each bidder, otherwise the substitutability and complementarity relationships would not be regarded. Suppose we want to distribute m tasks ($m = |G|$) to a number of agents that have published their bids. Let $v(S)$ be the cost for executing bundle S. The winner determination problem can then be formalized as an integer programming problem in the following form:

$$\begin{aligned}
\text{minimize} & \sum_{i \in N} \sum_{S \subset G} v_i(S) x_{S,i} \\
\text{subject to} & \sum_{S \ni j} \sum_{i \in N} x_{S,i} \leq 1 \forall j \in G \\
& \sum_{S \subset G} x_{S,i} \leq 1 \forall i \in N \\
& xS, i = \{0, 1\} \forall S \subset G, i \in N
\end{aligned}$$

Thereby, $x_{S,i}$ is a boolean variable that encodes if the bid of bidder i for the bundle S is part of the optimal allocation.

As the winner determination problem can be formulated as an integer programming problem, which is known to be NP-hard, it is NP-hard, too. It has also been shown that the winner determination problem is a set packing problem, which is in fact NP-hard, see Ruß and Vierke [RV99] or Shoham and Leyton-Brown [SLB09, p. 350].

To extend combinatorial auctions against strategic bidding extensions based on the idea of second price auctions, as already mentioned, in the description of the Vickrey auction have been discussed. This mechanism is named the Vickrey-Clarke-Groves mechanism, which is detailed out, for instance, in Wooldridge [Woo09, pp. 308–310], and Shoham and Leyton-Brown [SLB09, p. 346].

The idea of combinatorial auction has although applied to other problems, like scheduling. In scheduling, as already outlined in Section 2.2, the question of who performs the task is not the only question. Moreover, the question when the action is performed has to be dealt with. An idea to handle this challenge is, for example the usage of price vector that specifies the prices for different points in time, see Stockheim et al. [SSWG02]. Another approach by Elendner [Ele04] reformulates the problem to apply combinatorial auctions to scheduling tasks. Not the tasks are auctioned, but rather the time slots of the resources are objects in auctions.

The idea of combinatorial auction has attracted a number of economic researchers that use this concept in a number of different task allocation problems among autonomous entities. An example that has been mentioned by a number of researchers are supply chains, e.g., the work by Schmidt [Sch99] or Schwind [Sch07]. Combinatorial auctions have also been adapted by researchers from AI, e.g., Wellman et al. [WWWMM01].

Having in mind the goal of this thesis, the coordination of planning entities, at a first glance the idea of combinatorial auctions is appealing. Autonomy of the bidders is not effected in combinatorial auctions, only the necessary information is published to one entity, and the outcome is maximizing the social welfare. Nevertheless, there are some fundamental criticism against the usage of combinatorial auctions. The main argument is the complexity of combinatorial auctions. As already mentioned, the winner determination problem is NP-hard. But even worse, to generate the bids an agent has to evaluate the consequence of adding the items in the bundle to its current local plan. That is, a bidder has to generate $|\wp(G)|$ of bids. For a set of n goods the bidder has to compute $2^n - 1$ bids[17]. This in consequence means that the bidder has to run its planning system $2^n - 1$ times to generate the potential plans that are required to derive its bid. As we have seen in Section 2.2 these planning problems are typically NP-hard, or worse, as well. Consequently, conducting a combinatorial auction for the coordination of planning entities for n goods and m bidders would result in solving $(2^n - 1) * m + 1$ NP-hard problems. Thus, combinatorial auction are for the coordination of planning entities not computational efficient.
Current investigations by Gujo et al. [GSVW07] investigate how efficient combinatorial auctions work without the presence of all bids. That is, the resulting matrix of bids would not contain an entry in every cell. But as far as they author is aware, no final results of this research are available up to now.[18].

An introduction into the filed of auctions from a DAI perspective is given by Wooldridge [Woo09, Chap. 14] and Shoham and Leyton-Brown [SLB09, Chap. 11.2].

[17]As the empty set is not regarded as a bundle it is $2^n - 1$ instead of 2^n bundles resulting from n goods.

[18]According to Michael Schwind co-author of the aforementioned article (e-mail communication 02/10/2010).

Bargaining In this section we deal with bargaining, which can be described as the game theoretical analysis of formally described negotiations. According to Wooldridge [Woo09, p. 315] a bargaining situation can be described by:

- a negotiation set, representing the possible proposals an agent can make,
- a protocol, in game theoretical sense presented above, which defines the legal proposals in a given situation,
- a strategy profile, whereby the strategy of an agent is a private information, and
- a rule that determines the termination of the bargaining process.

Based on this formal description of a negotiation different bargaining situations have been investigated. Typically, the goal is to investigate what properties a mechanism have, and what strategies are implied by a given mechanism. While some of these settings are kind of artificial or simplified, other approaches have found a number of applications in the field of DAI. Here some of these analyses are discussed, see Wooldridge [Woo09, Chap. 15] or Sandholm [San99, Chap 5.5] for more details.

Bargaining for resource division If the task is to divide a resource among agent it is assumed that there exist two rational players how want to divide the resource. If they cannot come to an agreement they will end up in a conflict deal, which is the worst outcome of the bargaining for both of them. If the negotiation has only one round this game is termed an *ultimate game*. The first player offers a deal and the second one has to accept it, or take the conflict deal, which is the worst outcome. Consequently, the first player can choose an offer that maximize its profit, which results in taking the resource exclusively. This characteristic still exist if the number of rounds of the game is increased but finite. This can be shown by induction, for a proof, see Wooldridge [Woo09, p. 318]. Consequently, the bargaining process can be cut short in a one round bargaining process. The first agents offer to take the entire resource, and the second one has to accept. Of course, this result relies on the assumption that the players are aware of their counterpart's strategy. The game can be changed, if a time discount factor is taken into consideration. This will lead to the Rubinstein bargaining result, described, for instance, in more detail by Sandholm [San99, p. 222].

Bargaining for a price Of course, the assumption that each agent knew the strategy of each other, and thus, can shortcut the negotiation outcome is a strict

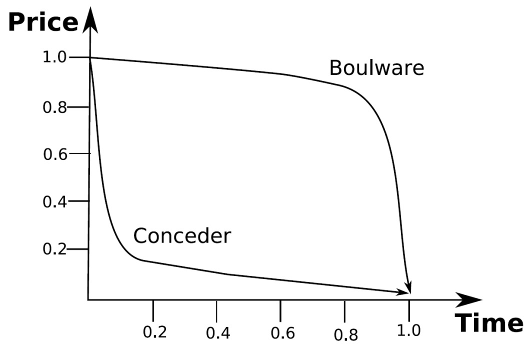


Figure 3.7.: Classic seller strategies, according to [Woo09, p. 322]

assumption. Another idea is that each agent has its *negotiation decision function* that describes how the agent makes concessions while the bargaining goes on. Thus, it is not necessary for the agent to know or take care of the strategy of the other players. Its strategy is encoded in this function. Wooldridge presents two typical negotiation decision functions the conceder and the boulware strategy. Both strategies describe different approaches how the willingness to make concession evolves over time. Examples are shown in Figure 3.7. These bargaining models are used to predict the outcome of a bargaining strategy. In experiments presented by Fatima et al. [FWJ05] particular these strategies have been investigated. Especially how their outcomes is effected by the agents apply and how much information they can use.

Bargaining about actions Another well-known investigation concerning the design of mechanisms that can be used to coordinate the activities of agents has been presented by Rosenschein and Zlotkin [RZ98b]. These authors define three different problem domains, these are:

- Task Oriented Domains: In a task oriented encounter a number of agents have to perform a set of tasks. Each agent has its own subset of tasks to perform. Each agent is capable of executing each task. By execution of each task costs are generated. These costs can depend on the agent performing the task and the other tasks this agent has to perform. Tasks can be exchanged among agents. The goal is to find an efficient task allocation among the agents.

- Goal Oriented Domains: In a goal oriented encounter each agent has a set of goals that he want to achieve. A goal is a state of its environment. Performing an activity requires limited resources and may influence the state of the world towards other goals of other agents. Thus, dependencies and conflicts can arise between agents. It is also possible that the agents try to satisfy most of their goals, in case an existing conflict hinders them form fulfill all their goals.

- Worth Oriented Domains: In a worth oriented encounter the agents are guided by an utility function that they try to maximize. Thus, it is possible that a goal can be fulfilled to a certain degree. The usage of an utility enables the designer to model the fact that goals can be partially achieved, instead of binary modeling in goal-oriented domains.

Rosenschein and Zlotkin provide a well-known protocol that can be used to bargain in such domains, the *monotonic concession protocol* [RZ98b, pp. 40,41]. According to this protocol the negotiation goes on for a number of rounds. In each round the agents propose a deal simultaneously. If an agents accept a deal the negotiation end. It accepts, if the utility of the received offer exceeds or is equal to the utility of its last proposal. If no agreement could be achieved in a round the next round starts. Thereby, no agent is allowed to offer a deal that has a lower utility for the other agent, as it has already offered before. If in a round (that is not the initial round) both agents make no concessions the negotiation ends with the conflict deal. For this protocol Rosenschein and Zlotkin present an appropriate strategy that is designed to be efficient, stable, and simple. They name of this strategy is *Zeuthen strategy* [RZ98b, pp. 43,44]. According to this strategy an agent should start with his local best deal. The concessions he made depends on the risk the agent wants to take ending in the conflicting deal. Risk is influenced by the distance from the current offered deal to the conflicting deal (in terms of utility). If this distance is low an agent is more risky then otherwise. The risk of an agent i for step t can be computed as follow:

$$\text{Risk}_1^t = \frac{\text{utility agent } i \text{ loses by conceding and accepting } j\text{'s offer}}{\text{utility } i \text{ loses by not conceding and causing conflict}}$$

In each step an agent computes its risk and the risk of the other agent. If its own risk is smaller or equal to its opponent's risk he should make a confession. Thereby, he should make a minimally concession, i.e., a concession that is just big enough that his risk becomes greater than the risk of its opponent.

In their work Rosenschein and Zlotkin extend the bargaining mechanism to handle conflicting situation and mixed deals, i.e., deals with a random influence, which leads to the Unified Negotiation Protocol [RZ98b, pp. 119–121]. Furthermore, these authors investigate different option how agent could behave strategically to gain an advantage and how this can be avoided to a larger extend by modifying the bargaining mechanism. Their work is a first and important step into the research for re-use of coordination mechanisms. They define characteristics of coordination problems that define a sub-set of coordination problems and present efficient coordination mechanisms for those sub-sets.

The contract net protocol, already been introduced in Section 3.1.2, has been extended for more elaborated, non-cooperative scenarios, as well. Sandholm [San96] defines a semantic of the protocol that makes it possible to apply this protocol for self-interested agents, as well. Note originally the contract net protocol was defined for cooperative agents, only. Thereby, the main idea while generating a bid is to compute the marginal costs of an additional task. Thereby, the marginal costs are the costs that occur additionally, by adding the offered task to the current set of tasks. Moreover, agents can suggest a re-allocation of tasks/resources. They can suggest four different types of contracts:

- original-contracts: describing one allocation to another agent,
- cluster-contract: specifying a set of allocations atomically to another agent,
- swap contracts: a pair of agents swaps a pair of tasks, and
- multiagent contracts: more than two agents exchange allocations atomically.

If all other participating agents accept, the change of a contract is adapted to the overall system. These types of contracts and corresponding negotiation schemes enables the agents to improve their situation constantly. An arbitrary sequence of such contracting operation can improve the overall system's performance, i.e., it can stuck in local optima. In fact Sandholm showed that with more complex contracts it is possible to come to a global optimal outcome with a finite sequence of individual rational re-allocations. This is possible with the OSCM-contracts that allow to suggest a complete reallocation of jobs plus the specification of side-payments from one agent to another. But it has to mentioned that the computational effort to compute such contract suggestions can become very complex. A more elaborated discussion about Sandholm's findings can be found at [San96] or [San99, pp. 236,237].

Based on the idea of commitments Sandholm and Lesser [SL02] discussed in their article the advantages of non-binding commitments. Announcing to break up a given commitment is called *decommitment*. In case an agent decide to decommit from a given commitment he has to pay a predefined penalty. These penalties enables to gradually adjusting the binding character of a commitment. Thus, these types of commitments are called *leveled-commitments*. Sandholm and Lesser show that using leveled commitments enables multiagent systems to find more efficient deals, in comparison to approaches where only binding commitments exist.

3.2. Agent-oriented Software Engineering

Agent-oriented software engineering (AOSE) is a research field connecting DAI and software engineering. Although one has to state that most contributions in this field are made by researchers from AI. Like in conventional software engineering, AOSE investigates issues concerning the different process steps of software engineering, see Sommerville [Som06, Chap. 4]. These are the specification of system's behavior, modeling systems, and supporting implementation of systems by methodologies, frameworks, and tools; and finally verification and validation of implemented systems. In the following the field of AOSE is surveyed and thereafter efforts and ideas for reusing components in particular coordination mechanisms are discussed.

3.2.1. Surveying agent-oriented software engineering

In the following we discuss results in the field of specification and design of multiagent systems. An important result, probably the most mature results of AOSE are frameworks to build multiagent systems. Finally, techniques for verifying and validating multiagent systems are discussed. A very detailed introduction into this field is given by Gómez-Sanz et al. [GSGW04].

Designing multiagent systems

In the specification of a system the expected behavior of the system is formalized. The elicitation of these specifications is typically done during the requirement elicitation phase. The field of requirement engineering is a subfield of software engineering. The metaphor of agents has been used in conventional requirements engineering, too. An agent in requirements engineering is "an active system component playing a role in goal satisfaction" [Lam09, p. 396]. With an agent-based modeling approach the responsibilities of the systems can be modeled. These models can answers the question "who is doing what and why" [Lam09, p. 395]. Nevertheless, van Lamsweerde argue that for most projects a goal orientation is more fruitful, as it aims at the questions what goal the entire system should achieve. A more detailed discussion is provided by van Lamswerde [Lam09, Chap. 7].

The integration of agents into requirements engineering has been surveyed by Gómez-Sanz et al. [GSGW04, pp. 36,37]. More recently Fuentes-Fernández et al. [FFGSP06] have presented a specialized approach or requirement elicitation in AOSE. It enables to acquiring the required information based on an agent-based model, based on ideas of activity theory. In their paper Fuentes-Fernández et al. transfer the idea of activity checklist to develop guidelines for the requirements elicitation process. Therefore, structured interviews are used, and an UML profile describing activity theory concepts has been developed by the authors. Based on that means a process for the requirement elicitation is proposed. Thus, this approach enables formal and informal, i.e., textual modeling of requirements.

Another way to specify the desired behavior of agents bases on formal specifications. The logic formulation might not be well suited for an elicitation process but rather for experts who want to assure that the system satisfy some behavior. An example for such a specification language is the situation calculus that is, e.g., the foundation of the ConGOLOG programming language for agents, presented by Giacomo et al. [GLL00].

Another example is the formal languages presented by Dastani et al. [DJT01]. These authors have presented a hierarchy of languages for specifying the dynamics of multiagent systems. In contrast to *conventional* systems the systems behavior is specified from different perspectives. Dastani et al. [DJT01] present the specification from a global, an agent's, and an environmental perspective.

A number of design methodologies have been proposed. Those methods aiming to structure the specification and design process of MAS. Examples for such methods are Gaia, Tropos, MASSIVE, Zeus, MaSE, and Aalaadin. In the following only some key concepts of these modeling methods are presented. A detailed introduction and comparison of these methods can be found in Weiss and Jakob [WJ05] and Bergenti et al. [BGZ04]. Central concepts for the design of multiagent systems are roles, interactions and organizations.

- **Roles** A role is an abstraction of a component of the system, see Caire et al. [CCG$^+$04, p. 179]. Roles have the notion of social rules, or job descriptions. According to Wooldridge et al. [WJK00] it can be described by sets of permissions, responsibilities, activities, and interactions. An agent can have one or more roles at the same time.

- **Interactions** Define the allowed interactions between roles. An interaction definition has to specify the interaction protocols the roles apply (see Section 2.3.4). Moreover, the goals of the interaction and required input are specified. Interactions define the communicative behavior among the different components/agents in the system.

- **Organization** The relations between roles can also be defined using organizational patterns. Roles are arranged in an organization. This enables grouping of competence and structuring the agent system.

Applying and extending UML diagrams The use of UML diagrams or extension of them has attracted a number of researchers, e.g., Depke et al. and Odell et al.

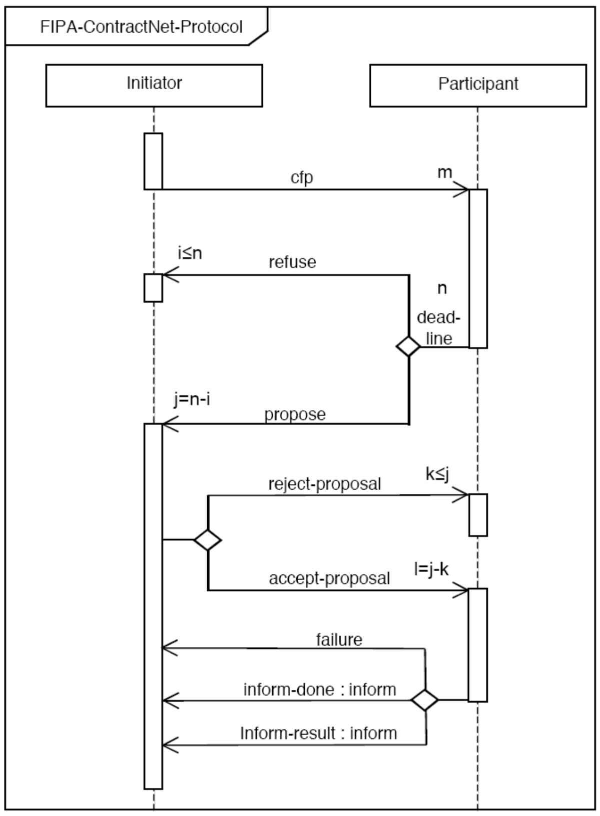


Figure 3.8.: AUML diagram of the contract net protocol [FIP02f, p. 2]

[DHK02, OPB01]. In particular we present here the probably most prominent approach that has been used in the FIPA specifications as well, the *Agent UML* (AUML) [OPB01]. Within AUML class diagrams and sequence charts from standard UML has been identified as valuable tool for designing agent-based systems. AUML extends UML1.4[OMG01]. UML 1.4 was the standard UML specification about the time AUML was designed [OPB01]. Especially sequence charts have been investigated as a mean to specify and design agents interactions [OPB01]. An example for such an extended sequence chart is shown in Figure 3.8. Currently, the development of AUML has been slowed down, as a consequence of the release of UML 2.1 specification [OMG07], that incorporate aspects of AUML [Com07]. For that reason, standard sequence diagrams in UML 2.x can be used instead of AUML sequence charts, today. The entire efforts among AUML are summarized by Huget et al. [HOB04].

Languages and frameworks for agent-based systems implementation

For the implementation of agent-based systems a number of programming languages from different programming paradigms have been considered. Among others declarative and functional languages like Mozart, Lisp, Prolog and Concurrent-METATEM are mentioned by Gomez et al.[GSGW04, pp. 55,56].

Moreover, specialized programming languages for agent-based systems have been designed, like Agent0, AgentSpeak, which is the foundation of the Jason framework by Bordini et al. [BHW07]; or ConGOLOG by Giacomo et al. [GLL00].

Agent-oriented development frameworks A lot of effort of research in the field of AOSE has been devoted to the development of agent-oriented development frameworks. In fact there exists a huge variety of frameworks ranging from frameworks based on logic-based agent programming, like Jason or Jadex, to frameworks extending object-orient paradigms concepts on top of object oriented languages, like JADE[19]. An overview about some of these frameworks can be found in Weiß and Jakob [WJ05] and Bergenti et al. [BGZ04]. Those frameworks offer the infrastructure in which each single agent can operate in. For instance, those frameworks provide means for communication between agents. Some of these frameworks are compliant to the aforementioned FIPA standard, which ensures a certain level of interoperability. As agents from different authors can communicate and cooperate within those environments. Most of these platforms, even if they are not completely FIPA compliant, take advantage of the FIPA abstract architecture standard [FIP02a]. These frameworks provide the agent environment discussed in Section 2.3.4.

Omicini et al. [OOR04] characterize those infrastructures for agents into two groups: the *enabling* infrastructures and the *governing* infrastructures. The enabling infrastructures provide infrastructure mainly for communication between agents. These infrastructures enable agents to exist in a MAS. According to Omicini et al. most of the existing infrastructure fall in this category. In contrast Omicini et al. argue that a governing infrastructure would offer additional services, like coordination (see Section 3.1.2). Thus, these infrastructures would govern the interaction space of the agents and offer a higher level of abstraction for the developer.

From model to implementation A current trend in software engineering is the model-driven development (MDD). The idea of this approach is to model the system and then automatically transform the model into executable code. In the

[19] `http://jade.tilab.com/`, Accessed: 03/15/2010

following research taking advantages of model transformation and automatic code generation for agent-based systems is presented.

Some agent-specific modeling tools offer code generation, e.g., the *Zeus-Tool*, offers a number of templates that could be configured by the developer to generate code. Other tools, e.g., *agentTool*, can generate method bodies that have to be implemented by the developer.

Another line of research wants to take advantage of the generation of agent code applying principles of model-driven development (MDD). An introduction into the field of MDD is given, e.g., by Stahl and Völter [SV06, Chap. 2]. Multiagent systems are specified on a meta model level. This meta model can be mapped to different agent platforms, like JADE for instance. This enables the designer to develop models on a high level of abstraction, and then automatically transforms the model into executable code. Modeling languages have been proposed, for instance, by Hahn [Hah08], who presents a domain specific modeling language for MAS. The developer has to specify its system in this domain specific language, which is a platform independent model (PIM). Code generators can transform such a PIM with additional templates into a platform specific model, e.g., a JADE representation of the system. For more details see Hahn [Hah08] or Hahn et al. [HZWF10].

Verification, Validation and Visualization

If a MAS has been implemented in a formal way, like an implementation using the ConGOLOG language, or the system has been generated automatically based on formal model, e.g., by MDA techniques presented above, it is possible to verify the MAS. By verification a formal prove is given, that the system will behave within its specifications. An interesting approach towards the verification of MAS is presented by Brazier et al. [BCG$^+$98] using the idea of hierarchical decomposition of systems for verification. Thus, to proof the specification for the entire systems it is decomposed into its subsystems. For each subsystem specifications are defined that, assumed they are true, would contribute to the proof of correctness of the overall system. This process is iteratively repeated for all subsystems until the considered subsystems have reasonable size to use conventional verification techniques. An introductory into these verification techniques is, e.g., presented by Apt and Olderog [AO94].

Other approaches assuming a complete specification of the MAS to be present, e.g., Bulling and Hindriks [BH09] or Hilarie et al. [HSKF05]. If a specification of the systems behavior, or at least subparts like its communication behavior are provided agents modeled e.g in logics can be verified. An example with more practical value is, for instance, implemented in the agent development toolkit

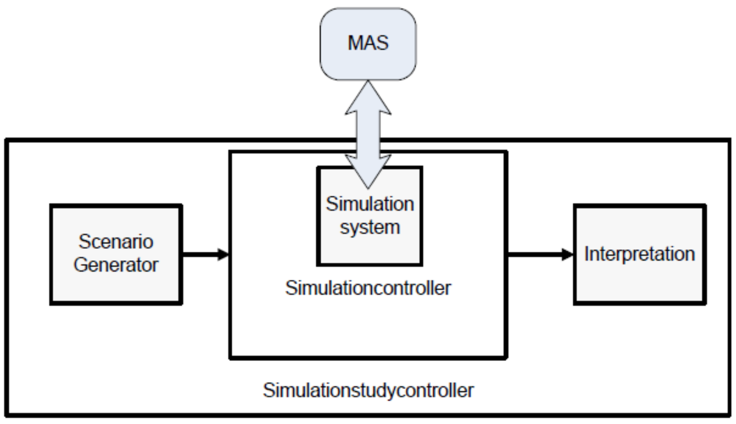


Figure 3.9.: Components of a simulation tool for validating MAS, [TS09]

agentTool. In *agentTool* conversations can be checked for problems, like deadlocks using the model checker *Spin*, see Weiß and Jakob [WJ05, p. 251]. Verification on the model level can be very useful during the design of the system, as design flaws can be identified upfront the implementation. If the system's implementation has not been automatically derived by verified translators, than it is not sufficient only to verify the model to gain any statement about the implemented system. In those cases the verification lacks a computational grounding.

Thus, it is hard to verify a MAS, either because of the lack of computational grounding of the formal model or because of the lack of a formal model. Thus what remains possible is validating the system. By validation we refer to a systematic testing the system and thereby increase confidence in the system.

Validation techniques often rely on testing and simulation as methodology. A comparison between conventional validation approaches known from software engineering and special needs and approaches from the agent-oriented software engineering have been compared by Fuentes et al. [FGSP04]. Systematic software testing has not been adopted widely in agent-oriented software engineering. Even though extension for conventional testing techniques, like unit-testing, have been proposed already, e.g., by Hormann [Hor06]. A systematic testing approach for the overall system has been discussed, e.g., by Fortino et al. [FGR05] and Timm and Schumann [TS09]. The idea is to test the behavior of the MAS in a synthetic environment that can be controlled by the tester. Thereby, the tester can change the environment in a controlled way, and observe the reaction of the MAS. An architecture for such a test environment has been proposed by Timm and Schumann and is shown in Figure 3.9.

The visualization of interaction of in a MAS can enable an easier understand of an existing agent system. It can also be used for debugging or testing issues. Visualizing communication can be realized by dynamically draw sequence charts, which is offered, e.g., by the *Sniffer Agent* of the JADE platform. More elaborated techniques for visualization, like Dooley graphs, have been proposed by van Parunak [Par96] or Singh [Sin98].

3.2.2. Reuse of existing methodology in AOSE

Research done in the field of AOSE has created a variety of methods. So, it is desirable to gain the entire benefit out of this large body of work by using existing mechanisms. This will lead to increased development speed and higher reliable software. For that reason existing work has been surveyed according to what concepts of reuse exist in AOSE. Therefore, the proceedings of previous workshops on agent-oriented software engineering have been surveyed: [CW01, WWC02, GOW03, GMO03, OGM04, MZ06, PZ07, LP08, LGS09]. This workshop series is an annual workshop co-located with the AAMAS conference[20]. The series started in the year 2000. So the results of this section base on a survey of the last ten workshops[21]. Moreover, additional literature was surveyed, like the proceedings of the International Conference of Software Reuse[22] and the International Journal of Agent-Oriented Software Engineering[23]. In the specific field of reuse in software engineering the field of agent-oriented software engineering has not been recognized at all, at least there exists no single paper concerning agent technology. The research concerning reuse in software engineering primarily focus on the selection process for commercial off-the-shelf (COTS) software. The international journal of agent-oriented software engineering started in 2007 and appears quarterly. The impression from the survey of the AOSE-proceedings, that reuse has not been identified as a specific topic of itself in the field of AOSE, get's supported by the survey of this journal. Up to now[24] the issue of reuse of methodology has not drawn much attention by researchers.

Before we go into further detail, two types of reuse have to be distinguished. Reuse of code and reuse of concepts. The first type of reuse can be found in agent-based frameworks/infrastructure, where a code base is common to a num-

[20]The AAMAS conference is one of the major annual conferences in DAI, organized by the International Foundation for Autonomous Agents and Multiagent Systems, for more details see `http://www.ifaamas.org/`, Accessed: 02/15/2010.

[21]At the time of writing, the papers for the 2010 workshop are under review.

[22]For an overview of the proceedings see `http://www.isase.us/pastconferences.htm`, Accessed: 03/08/2010.

[23]`http://www.inderscience.com/browse/index.php?journalID=174`, Accessed: 03/08/2010

[24]Current issue is 2010 (Vol 4., No.1), date 03/08/2010

ber of applications. The second type of reuse addresses concepts and mechanisms that can be reused, as well. An example might be interaction protocols, like the contract net protocol. Consequently, one can argue that reuse has been facilitated by the development of efficient frameworks for agent-based systems. In fact we focus here on the second type of reuse, the reuse of concepts and methods. Existing modeling approaches, presented above, might offer reuse of artifacts that have been created in previous designs or are pre-defined within a toolkit or framework. Although, various frameworks for building agent-based systems have been proposed and are in use. Within these frameworks concepts like agents, and messages and infrastructure aspects like message exchange is supported, but no specific means for coordination of agents in general or their plans in particular, are supported within those frameworks, typically.

An established way to reuse concepts in software engineering has been the definition of patterns, e.g., the well-known design patterns by Gamma et al. [GHJV94]. For the field of agent-oriented software engineering Lind [Lin03] suggested a format for agents oriented pattern and present an architectural and an interaction protocol, for example. Architectural patterns, like Broker, Moderator, and Wrapper have also been presented by Heyden et al. [HCY99]. Interaction patterns, like patterns for Subscription, Call for proposals, Broker, and Wrapper have been discussed by Kolp et al. [KDF05], as well. Those authors termed those patterns *social patterns*. The authors present a framework for describing those patterns in a unified way, that has been specialized for agent-based development. Those architectural and interaction patterns have been standardized by the FIPA. The FIPA standards have been presented in Section 2.3.4.

A wider scope of patterns in AOSE has been proposed by Sauvage [Sau04]. Sauvage distinct in his article between MetaPatterns, that describe abstract constructs for the design of agent-based systems. He introduces organizational schemes, like organizations and roles, and protocols as two meta-patterns. The second group of patterns identified by the author are so-called metaphoric patterns. Sauvage mentioned marks, like pheromones, and influences as two metaphoric patterns. The third class of patterns are according to Sauvage architectural patterns, addressing the architecture of agents, like layered architecture or BDI architectures, already presented in Section 2.3.2.

Design patterns for the design of self-organizing systems have been summarized by Gardelli et al. [GVO07]. The patterns presented by those authors are collected from the design of nature-inspired design of self-organizing systems, like ant-based systems, for instance, patterns addressing the evaporation, aggregation, and diffusion are presented.

The identification of particular design patterns for the coordination in self-organizing systems is addressed in the article by de Wolf and Holvoet [WH06]. The authors present two design patterns for coordination. These techniques are gradient fields and market-based control. The idea of the gradient field pattern is to model a artificial environment for the agent and to model the coordinated behavior by the movement within this artificial environment. Market-based control coordination techniques uses the concepts of prices to bring together producer and consumers of services or information, as a coordination mechanism. The authors aim in their pattern description on the problem of efficient allocation of task among agents.

An idea for reusing proofs within the validation of multiagent systems base on the idea of the component-based verification proposed by Brazier et al. [BCG$^+$98]. Thus, only the proofs for components that have changed have to be updated. If it is possible to proof that these subsystems stay within their previously defined specification, the proof of the overall systems specification remains valid.

Hilarie et al. [HSKF05] argue that to facilitate reuse of agents or components of agents, it is necessary to formally specify these components and then proof the compliance of components to their specification. This can foster reuse, as those components can become the building blocks for future systems. For that reason the authors present a formal notation combined out of Objective-Z and statecharts. The authors present a case study for components of a robotic system. The focus of the authors is on the design of the formal notation and on prove of compliance. To the best of the author no common catalog of components exists that specifies existing components. And even such a catalog exist this is not sufficient. The availability of a number of specified components does not enable reuse, as no process how to select components exists, see also the discussion in the next section.

A quite similar approach for the reuse of organizations has been proposed by Jonker et al. [JTY05]. In their paper the authors present a formalism to describe organizations and they assign properties to these organizations. They propose to build a library of organization forms. An organizational designer then should be able to place queries to the library and retrieve possible organization forms that might suit his requirements. Of course, the indexing of those organizations becomes critical. The authors propose different aspects for indexing, e.g., by group functionality, environment assumptions or realization constraints [JTY05]. The retrieved organization descriptions might be adapted to the given situation at hand.

Bartolini et al. [BPJ03] argue that the current representation of interaction protocols is not sufficient, for protocols like the English auction, as by current specifications (e.g., FIPA standard [FIP01c]) only the sequence of messages is fixed. But more information is required, e.g., to generate a valid bid in an English auction. A new bid has to be higher than the currently highest bid. This kind of information has to be implicitly encoded by the agent designer. Thus, for achieving a better and machine-readable format the authors present a framework for specifying negotiations, based on rules for encoding the negotiation protocol. Agent should be able, i.e., by rule-based systems, to reason about those protocols and apply them autonomously. The reuse is thereby on emphasizing a more precise and complete form of specification for negotiations. The authors argue that by more precise description the reuse is facilitated as the agents can choose an interaction protocol automatically from a given knowledge-base.

Reuse of coordination mechanisms

As previously mentioned in this study we are interested in the coordination of autonomous planning entities. As outlined in Section 3.1.2 research in coordination of agents has generated a large body of work, containing a lot of different coordination mechanisms.

An idea, already discussed, for the reuse of coordination mechanisms is the coordination artifact (see Section 3.1.2 or [ORV$^+$04]). To briefly re-iterate the concept, in a coordination artifact a coordination mechanism is embedded that can be used by the agents. The problem in defining those coordination artifacts is to find the right level of granularity. So, if an artifact is too general it cannot be applied by the agents, as it does not fit their purpose. On the other hand a too specialized coordination mechanism in the artifact cannot be reused as the coordination between two specific planning entities might be a unique situation. Finding an adequate trade-off between generalization and specialization is hard to find, if at all. Moreover, the resulting governing infrastructures (already discussed above) have not been established in the developer community and it is not clear today, if they ever will. The only operational agent infrastructure mentioned in the literature, is the TuSCoN infrastructure. Applications of this platform are published only by the developers of this particular platform[25]. So it looks like if there is no active community around that infrastructure.

The need for an easier retrieval for reusing interaction protocols has been identified by Bussmann et al. [BJW03]. Therefore, the authors focus on the selection process of interaction protocols. To be applicable an interaction protocol has to respect the existing dependencies of the current situation. Therefore, the authors suggest to classify interaction protocols according to a number of criteria.

[25]for details see `http://apice.unibo.it/xwiki/bin/view/TuCSoN/PublicationsWithTuCSoN`, Accessed: 02/15/2010

These authors suggest to use the number of agents involved, the computability of constraints and preferences, the number of agent roles, the role assignment, the number of joint commitments, and the size of joint commitment as criteria. An agent designer should specify its requirements according to these criteria and then identify an interaction protocol that might be suitable for the given situation.

3.3. Identification of research gap and goals of this thesis

We have outlined the field of coordination in DAI. The presented coordination mechanisms have been designed for a variety of applications and base on different theoretical assumptions. In the literature [SSWG02, Sch01, BCGZ01] different schemes for classification of existing coordination approaches have been proposed. For instance Stockheim et al. [SSWG02] characterize different coordination approaches according to the solution methodology they use. In particular they cluster existing approaches into:

- combinatorial auctions,
- bargaining processes,
- random search,
- knowledge-based systems, and
- learning systems.

As already pointed out, Schumacher [Sch01] presented a classification that uses the perspective of the coordination mechanism as criteria, leading to subjective and objective coordination approaches.

Moreover, Busi et al. [BCGZ01] presents a framework for models and languages for coordination. They point out that key issues for the classification of coordination mechanisms are:

- coordinatables,
- coordination medium, and
- coordination rules.

All these classification schemes reflect the fact that different techniques, perspectives and theories have influenced the research concerning coordination in DAI. But non of these classifications either point out any application oriented criteria, neither indicate what motivations, i.e., use cases, have influenced the coordination mechanism. So the current state of the art of the field of coordination in DAI contains a huge variety of techniques and existing classifications indicate more the origins and roots of these approaches instead of their fields of applications, their strength, and weaknesses.

Researchers like Durfee [Dur99] and Sandholm [San99] have surveyed principles and techniques that can be used for coordinating multiple agents. But they do not give indications when to use such techniques. Consequently, the value for re-use of coordination techniques is limited. Rosenschein and Zlotkin [RZ98b] have proposed specific characteristics for coordination problems and means that allow for an efficient coordination in those specific problem settings. The state-, goal-, and worth oriented domains, discussed in Section 3.1.3 are an example for such a classification. If a given coordination problem falls into one of those domains the coordination mechanisms identified by Rosenschein and Zlotkin can be used. In contrast to the focus of this work the presented domains are a limiting factor. Moreover, their classification is still abstract, and additional requirements cannot be integrated. Although no indications for an implementing process are given.

Another field that has been surveyed in this chapter, was the agent-oriented software engineering. After we had given a broad introduction into this field we have discussed research concerning re-use and mechanism idenfification/selection in the field of AOSE.

A result of this survey is that the idea of reusing existing concepts is not very wide-spread in AOSE, which is underlined by the relatively rare rate of articles published addressing this issue. Most of the research described in the literature investigates techniques or tools concerning the development of new multiagent systems from scratch. The integration of an agent-based solution into an existing software ecosystem seems to be an open research fields. Similar conclusions have to be drawb for the rest of the life-cycle of an application, aspects like the maintenance and evolution of existing agent-based systems are not sufficiently covered.

The identification/selection process for a coordination problem has been tackled on the one hand by researchers who argue that the selection of a coordination mechanism should be done at runtime by the agent itself, e.g., by Excelente and Jennings, and Omicini et al. [ETJ04, ORV+04]. But in their work only toy-world-examples are presented, where the reasoning of the agent concerning existing dependencies is uncomplex. These approaches cannot scale to complex coordination problems, like the coordination of autonomous planners.

On the other hand the work from Bussmann et al. [BMR+02] suggest to classify interaction protocols and build a catalog of interaction protocols, similar to the approach presented for organizations by Jonker et al. [JTY05]. But the classification suggested by Bussmann et al. is not sufficient for coordination mechanisms because they remain too vague. The approach might be appropriate for the pure identification of interaction protocols, but coordination mechanisms are on the one hand only a subset of interaction protocols and on the other hand comprise more than simple rules concerning message exchange, which has also pointed out by Bordini et al. [BHW07] and Ossowski [Oss08].

From an engineer's point of view, who wants to find an appropriate coordination mechanism for a coordination problem consisting of inherently distributed planning entities, the current state of the art in the research of coordination and agent-oriented software engineering is unsatisfactory. The identification/selection problem of a coordination mechanism, in general and especially for the coordination of autonomous planning entities, is an open question in agent-oriented research. There exists neither tooling nor process for the identification/selection and implementation of coordination mechanisms that could be used for the coordination of planning entities.

This constitutes the issue addressed in this thesis, the effective identification/selection of appropriate coordination mechanisms for the coordination of existing planning systems embedded in autonomous planning entities. Or to rephrase it as the research question of this thesis:

> How can existing coordination mechanisms be efficiently identified, that are suitable for the concrete coordination problem of existing planning systems, in inherent distributed planning environments?

To answer this question it is necessary to define a process. Gaitanides defines a process as the "structure for actions" [Gai04, p. 1212]. The process has to structure the necessary actions that are needed to identify the suitable coordination mechanisms, that are applicable to the given problem. Moreover, it is necessary to detail out the process to provide means for a selection among those mechanisms. To establish a process three steps are necessary:

- process definition
- process validation
- process evaluation

In the first step the process has to be defined and detailed to be executable. In the second step the process has to be validated. This is to ensure the feasibility of the process. In the final step the process has to be empirically evaluated on a sociologically sound base. In this thesis we focus on the first two steps, the process definition and validation. In the next chapter we define and detail out the ECo process that is suggested for the efficient selection of a coordination mechanism.

In chapters 5 and 6 we present the validation of the ECo process applying it to two cases studies. Within these case studies we show the feasibility, flexibility, and power of the process that is proposed in this thesis.

4. The ECo-CoPs approach

As pointed out in the last chapter, the goal of this thesis is to define a process that enables the identification of a coordination mechanism for existing inherently distributed planning entities. For this reason we introduce the ECo process in this chapter. The ECo process consists of the following five steps.

1. modeling the scenario,
2. defining coordination requirements,
3. identifying suitable coordination mechanisms,
4. implementing coordination mechanisms, and
5. evaluating coordination mechanisms.

We detail out the ECo process and required process steps in Section 4.2. The ECo process is supported by the CoPS process and the CoPS framework, both presented in Section 4.3.

But at first we propose a set of six characteristics of coordination problems that could be identified in a number of coordination problems and that are outlined in the examples used in this study. Moreover, the presented coordination mechanisms are classified according to these characteristics. This enables a designer to identify suitable coordination mechanisms by matching required and provided characteristics.

Using this first classification scheme for coordination techniques enables a first and fast reduction to a subset of coordination techniques, presented in the previous section. But this is not sufficient for the selection of an appropriate coordination mechanism. The goal of coordination is to ensure that the different plans of the planning entities can be executed in a feasible way. Moreover, the coordination mechanism should be applicable in the given context. This is defined more precisely in this chapter. The evaluation of the results of the coordination process remain domain specific. Thus, we are not interested here in defining a specialized coordination mechanism or a family of those, that lead to efficient coordination mechanisms for a specific number of scenarios, e.g., by removing all redundancy in a joint plan. Moreover, we are interested in identifying suitable existing coordination mechanisms for a given situation and to enable a fast prototypical implementation to evaluate the coordination mechanisms in the given situation.

To address this issue, a set of activities has to be performed in a structured way. By suggesting the ECo process as a mean to do so, we offer a solution for the identification process for coordination mechanisms. One step of the ECo process is the implementation of the coordination mechanisms, to perform a quantitative evaluation. To support this step and to address the issue of implementing coordination mechanisms in an existing software environment, we present the CoPS process and the CoPS framework. In the CoPS process, an optional sub-process of the ECo process, the activities for implementing coordination mechanisms are discussed and structured. For an easier and faster implementation, the CoPS framework has been designed to support the CoPS process. The CoPS framework enables the designer to use pre-defined software agents that can be adapted according to the selected coordination mechanisms and to the local policies of the planning entities. While designing the ECo and the CoPS process, we aim to allow the designer an independent usage of these processes and the CoPS framework, as far as possible. It is possible to apply the ECo process for the identification/selection and to use other implementation mechanisms. It is also possible to use the CoPS process implementing a coordination mechanism, that has been selected in a way, that may not conform with the ECo process. Especially, the CoPS process has been designed in a way that no unnecessary restrictions for the agents, their architecture or abilities are implied. For that reason we discuss modeling aspects with formal and informal means. A formal specification enables agents with reasonable inference capacities to reason about appropriate actions by themselves. But this reasoning within agents should not be mandatory to build agents to coordinate the planning processes, as this task is orthogonal to the agent's architecture. Thus, providing an informal description for agent developers, who encode these policies implicitly into the agents, can foster the development of these agents. Moreover, it is possible to replace the implementation framework or design an agent for coordination, even without any framework. This modular approach facilitates independent use and independent development of all parts of the ECo-CoPS approach.

The ECo process, as well as the CoPS process and framework is presented here in detail. A validation of this process and the CoPS process and framework as supporting means is presented in the next chapters.

4.1. Characteristics of coordination problems and techniques

In this section we point out some characteristics of coordination problems arising between planning systems, that can be identified in the examples used in this study. We define these characteristics in a way that a simple, binary answer could be given to point out important characteristics of a coordination problem. These characteristics guide the identification/selection process for a coordination mechanism to efficiently coordinate the autonomous planning entities. The questions to identify the six characteristics are presented in the following. Possible answers are shown in brackets.

- Is an **allocation** problem part of the coordination, i.e., have tasks top be allocated to planning entities? (Yes/No)
- Are the local objective functions comparable, i.e., is the same measurement used by different planning systems? (Yes/No)
- Are the planning systems involved homogenous or heterogeneous, i.e., are the planning problems of the same type or are they different? (Homogenous/Heterogeneous)
- Does a common objective function exist that can be used to measure the success of the overall system? (Yes/No)
- Is information hiding necessary among the planning entities? (Yes/No)
- Do cyclic dependencies exist? (Yes/No)

Characterization of the examples

In the following we discuss the characteristics of the exemplified coordination problems used in this study. The result of this discussion is summarized in Table 4.1. Again, the example of production scheduling and distribution of goods (introduced in Section 2.1.1) is referred to as SPT. SCM is the common abbreviation for supply chain management, and CTM is the abbreviation for container terminal management. Obviously, within the SPT problem there exists no task allocation problem among the different planning entities. Each product has to pass the three areas in the same linear sequence, like in a flow shop scheduling problem. Even though the packing and the transportation planning try to reduce costs, this is not true for the scheduling system, which tries to optimize the utilization of resources. Consequently, the objective functions of the planning systems are not comparable. The planning problems are heterogeneous. All these three subsystems are part of

Characteristics	**SPT**	**SCM**	**CTM**
Allocation	No	Yes	No
Comparable objective	No	both possible	No
Homogenous/Heterogeneous	heterogeneous	both possible	heterogeneous
Overall objective	Yes	Yes	No
Information hiding	No	Yes	No
Cyclic dependencies	No	No	Yes

Table 4.1.: Characteristics of the SPT, SCM and CTM problem

the production and delivery process of one company; consequently, there exists an overall objective function, i.e., the costs incurred for the company. With a similar explanation it can be argued that no information hiding is necessary. And as discussed above in the simple linear case of the SPT, no cyclic dependencies exist.

The SCM (supply chain management) problem has been presented in Section 2.1.2. An allocation becomes necessary if companies have overlapping competencies, which is, according to Corsten and Gössing [CG01], rather frequently. Consequently, there exist more than one entity, here company, that could perform a task. It can be assumed that all companies want to maximize their profit. On a more detailed level it can be challenging to compare all local plans with a common metric. Thus, it depends on the concrete scenario and its modeling if the objective functions are comparable. The same is true for the type of planning problem. From an abstract point of view all planning systems can be abstracted as scheduling problems. Some abstract activities have to be performed and each requires an amount of time. Of course, the concrete planning problems of a shipping company deviate from the planning processes in a ware house or plant. It is reasonable to assume that the overall objective of the supply chain can be measured by an overall objective function. For instance, final goods produced by the supply chain that are sold to customers and the production and transport of these goods have induced costs. The difference between revenue and costs can be defined as profit of the supply chain, not regarding how this profit is split among the companies in the supply chain. This is typically a question of market power or more analytically the outcome of a cooperative game. As already mentioned, companies within a supply chain have overlapping competencies. Thus, a confidential level is requested by the companies and information hiding becomes an issue. Typically, it is assumed that supply chains have no cyclic dependencies, the flow of material is assumed to be linear. This is, of course, an assumption that does not necessarily have to be true, for instance, for supply webs the flow of material is not linear anymore and cycles could occur.

Finally, the characterization of the CTM (container terminal management) problem is discussed. As in the SPT there exists no allocation problem. All problems have to be solved to serve a ship. The used objective functions are not comparable. For instance, the solver for the berth allocation problem wants to achieve a high utilization of the berth and low waiting time for the ships. The crane scheduling solver aims to reduce the process times of ships and maximize the utilization of its resources. The planning problems are heterogeneous. Even if the direct monetary effects of all planning decisions could be estimated for the container terminal, this is not sufficient for achieving an overall objective function, because the service time for ships is a competitive factor that can hardly be estimated in monetary terms. Therefore, a trade-off has to be made between offered service time to ships and induced costs. Consequently, no overall objective function could be defined easily. All planning problems occur within one company, thus no information hiding is required. As shown above in Figure 2.11 (on page 35), there exist a number of cycles among the planning problems.

Categorization of presented coordination mechanisms

In Section 3.1.2 a number of different techniques for coordination among agents plans have been presented. Thereby, we gave a broad overview of existing techniques developed in the field of DAI. This overview is not exhaustive in the sense that all existing variants of all mechanisms have been presented. But it was intended to cover the wide field of coordination in DAI. In this section the suitability of these concepts for the coordination of planning systems is discussed. Therefore, we use the characteristics discussed in the last section. The findings of this discussion have been summarized in Table 4.2.

In this table we have listed all groups of coordination mechanisms presented in the previous discussion (see Section 3.1.2). Thereby in each row an entry represents a type of coordination mechanisms, and there could exists a number of mechanisms that can be classified as a member of a particular group. For instance, the type *negotiations* covers various different mechanisms that have been presented in the literature.

We have investigated the necessary characteristics for each of these mechanism groups, i.e., what characteristics are necessary to apply the coordination mechanism. If the mechanism is robust against the different types of a characteristic this is indicated by an asterisk (*). Rows that are stroke out indicate that this mechanism is classified as not suitable. This table is, of course, not sufficient to pick a specific coordination mechanism, but it can guide the search for appropriate mechanisms by narrowing down the list of possible candidates.

Mechanism	**allocation**	**comp. obj.**	**homo./hetero.**	**overall obj.**	**inf. hiding**	**cyclic dep.**
task Sharing	Y	*	*	*	*	*
contract net	Y	*	*	*	*	*
auctions	Y	Y	*	*	*	*
result Sharing	N	*	*	*	N	N
GPGP	*	*	*	Y	*	?
negotiations	*	*	*	*	*	*
centralized planning	*	Y	homogenous	Y	N	*
dpcp	*	*	*	*	*	N
dpde, preplanning	N	*	homogenous	*	*	*
dpde, plan merging	*	*	*	Y	N	*
dpde, mediator	*	*	*	Y	(N)	*
dpde, iterative	—	—	—	—	—	—
DCSP	—	—	—	—	—	—
coordination artifacts	—	—	—	—	—	—

Table 4.2.: Suitability of mechanisms according to characteristics; decentralized planning and execution (dpde), decentralized planning for a central plan (dpcp)

Task sharing, as the name suggests, is an approach to distribute tasks among a number of agents. Thus, it is primary suited for coordination problems involving a number of tasks to distribute among agents. For all other criteria it does not depend on particular characteristics of the coordination problem. The same characterization can be made for the contract net protocol. Auctions are another approach to distribute work among a number of agents. But to use them, it is required that agents have a comparable objective functions. This is necessary to generate comparable bids.

The idea of result sharing is a priori not well suited to distribute tasks among agents. Protecting private information is generally not coherent with the result sharing approach. As discussed above, the existence of cyclic dependencies can be a contraindication for applying result sharing approaches, as they may lead to problems with the termination of the process. Additionally, it has to be mentioned that through the possible chaotic behavior of planning systems result sharing has to be used carefully.

In coordination mechanisms like GPGP an overall objective is required, that all agents try to optimize. This is typically expressed in terms of quality that the agents want to maximize while executing methods of the task structure. Information hiding can partially be realized by defining a strategy for the result sharing that can be used to formulate strict rules for publishing information and to give commitments towards other agents. Concerning the characteristic *cyclic dependencies*, up to now, no clear indications can be made from the case studies in the literature, to what degree cyclic dependencies can be handled by the GPGP coordination mechanism.

Negotiations are a general concept that can be adapted to a number of characteristics. Therefore, no restrictions could be identified that limit the application of this concept. Note that there exists a huge number of different negotiations that might have special requirements towards these characteristics.

The centralized planning for the coordination of different agents can lead to very complex planning problems. Therefore, we think this might only be reasonable for homogenous problems. We are aware that there exist approaches for integration different planning problems into larger models, e.g., by Meisel and Bierwirth [MB06] and Park and Kim [PK03] who have presented approaches for integrating the BAP and CSP problems in the CTM domain. But those integrated models are specific for a certain situation and cannot be extended easily. For instance, it could be desirable to integrate the interdependent storage space allocation problem, too. Thus, the integration of models does not scale from the design point of view, i.e., the efforts for integrating an additional partial model grow faster than linear. Moreover, the computational complexity of the resulting integrated problem growth and computation complexity also does not scale well for the integration approach. To guide the overall planning process, comparable objective functions and an overall objective function are required to estimate and compare different alternatives and select the most suitable one. As all data required for plan generation has to be collected in the centralized planner, information hiding aspects cannot be respected.

The decentralized planning for a centralized plan can handle most of the aforementioned characteristics. Nevertheless, as the typical process is a sequential plan refinement process this mechanism can hardly handle cyclic dependencies. As already discussed they could lead to a non-terminating coordination process.

For decentralized planning with decentralized execution different techniques have been presented and are discussed. Among others, negotiations have been proposed. As negotiations have been discussed already above, they are omitted here.

The generation of constraints in the pre-planning coordination approach assumes that the initial task assignment has been made already [SW07]. Thus, it cannot be used for task allocation. As already mentioned, the problem is complex so we do not think it is useful to increase complexity further by coping with different kinds of planning problems. Therefore, we argue it might be suitable for homogenous planning problems.

The idea of plan merging requires a lot of information about the existing dependencies among plans and about transformation of local plans that can result in executable local plans. Thus, information hiding requirements cannot be satisfied. Moreover, a global objective function is a prerequisite to have means to be able to choose among alternatives solving a conflict.

Mediated-based approaches require a global objective function to decide among alternatives in conflicting situations. In contrast to merging, the mediator does not necessarily need all details of the local planning problems and policies, which can make information hiding realizable. It is necessary to have comparable objective functions among agents, to generate a meaningful representation of the system's state and the current system's overall performance.

As already discussed, the coordination of planning systems cannot be achieved by any type of fix-point iterations. Moreover, the communication of partial results is not advantageous because of the chaotic behavior of planning systems. Thus, ideas of iterative plan formations seem not suitable for the coordination of planning systems.

The re-formulation of the coordination problem as a type of distributed constraints satisfaction problem is appealing, but it does not offer us to use existing planning systems and all problems have to be formulated as a constraint satisfaction problem and interdependencies have to be added as additional constraints. Techniques for distributed constraint optimization (DCOP) could also be used to find optimally coordinated plans. Nevertheless, if each sub problem is not given as a constraint satisfaction problem, which is the common situation, distributed constraint satisfaction seems not an appropriate approach. But the formulation as a DCSP or DCOP enables more detailed analysis of effects of different problem decomposition techniques. Thus, for a more strategic perspective the formulation of coordination problems as DCSPs can be an interesting research issue to investigate effects of different distributions of sub problems among problem solvers on achievable solution's quality.

The usage of coordination artifacts aims at externalizing the coordination knowledge and engineering out of the agent into the environment of the agents, or into the infrastructure the agent is executed in. This externalization might be advantageous for an easier design of agents, but it requires that all knowledge about the coordination mechanism and required information has to be given to the artifact. In the context of coordination of planning systems this would mean to externalize all possible plans of the agents, or to access to the local planning system to the artifact. Therefore, we are convinced that this approach is not suitable for the coordination of planning systems.

4.2. The ECo Process

In this section we present the ECo process. ECo is an abbreviation for *Engineering Coordination*. This process comprises an identification process of appropriate coordination mechanisms. It also covers the implementation and evaluation phase.

The aim of the ECo process is to guide the designer during the process of implementing automated coordination mechanisms for autonomous planning entities. Therefore, especially two steps are necessary: Selecting appropriate coordination mechanisms and implementing them in the given context.

Most research performed in software selection addresses the selection of commercial software, like ERP systems which is discussed, e.g., by Wei et al. [WCW05]. Another process model has been presented by Krcmar [Krc10, pp. 167–190] , as part of the management of information systems. Similar process descriptions can be found by various consulting companies, who have defined their process models, as well. But the resulting process descriptions are not applicable as neither request for information can be handed to a provider nor does it make sense to make cost comparisons or request presentations by vendors, as coordination between planning systems has no market, yet.

In the field of business informatics there exists a subfield of systems theory, e.g., presented by Krallmann et al. [KST07]. One subfield of this systems theory is system analysis. Within system analysis Frank et al. [FBH07, p. 135] present a standard process model for software selection processes that can be tailored for the selection of an appropriate coordination mechanism. In the standard process model the following steps are defined:

- project definition[1],
- model the current situation[2],
- define the goal situation[3],
- selection of appropriate mechanisms to transfer the current situation into the goal situation[4], and
- implementation of the selected solutions[5].

In the first step the goals of the project are defined. This phase is in general necessary in system analysis, but does not need to be transferred into the ECo process. The goal of the ECo process has been outlined clearly up to this point.

The next two steps are subject of the field of requirements engineering. A detailed introduction into the field of requirements engineering is provided by van Lamsweerde [Lam09]. The goal of these two phases is to achieve an understanding of the required characteristics and features of a solution. The third step concerning the selection of the appropriate concepts. In conventional system analysis the decision to make or to buy a solution to achieve the goal state is typically made in this process step. Due to the large number of possible approaches to achieve a coordination process, we propose that the selection process has to be divided into different steps. First, approaches that are applicable have to be identified. These approaches have to satisfy a number of strict coordination requirements. Then, a more detailed review of the remaining coordination mechanisms is possible. Stantechev points out that a final evaluation of the usefulness of a solution, here the coordination mechanisms, can only be drawn if the solution is implemented at least in parts of its addressed system [Sta07, pp. 290,291]. This can be an expensive task, and therefore we suggest in the ECo process the usage of prototyping for the generation of prototypically implemented systems [Som06, p. 410], to ensure a fast, but not necessarily computational most efficient version of a coordination mechanism. The coordination mechanisms in the prototypes might be inefficient, e.g., in terms of required time, send messages. Thus, the mechanisms cannot be implemented efficiently. The results of the coordination process do not have to be effected by the fact we are using a prototype. This prototypical

[1]German original term “Projektbegründung”
[2]German original term “Istanalyse”
[3]German original term “Sollkonzept”
[4]German original term “Realisierung”
[5]German original term “Implementierung”

implementation can then be quantitative evaluated. In this last step the quality of the coordinated plans is measured. Other aspects of the implementation, like the adaptation and/or the specification of conversation policies should only be done once in a proper way, as they provide the infrastructure for the coordination mechanisms. Within the implementation step the selected solution is realized. These process steps have to be detailed for the ECo process. Moreover, we think that it is not sufficient to end the process with the implementation phase. A control phase is necessary to evaluate if the goals have been achieved and to assess the solutions quality. This is necessary to evaluate if the expected results from the selection process can be really gained in the concrete environment, which is a recommended step in systems analysis of Frank et al. [FBH07, p. 185], too. In consequence, the ECo process model comprises five steps. These steps are:

1. modeling the scenario,
2. defining coordination requirements,
3. identifying suitable coordination mechanisms,
4. implementing coordination mechanisms, and
5. evaluating coordination mechanisms.

For the application of these process steps we propose an iterative execution model, similar to the well-known iterative software development process, like the spiral model or the iterative-waterfall model. For details of these software development processes we refer to the introduction by Sommerville [Som06, p. 65–74]. It enables to go to previous process steps and refine results of process steps if this becomes necessary. This is consistent with the perspective of system analysis that defines their process as iterative heuristic process with feedback [FBH07, p. 136]. The process model is depicted in Figure 4.1.

Note that it is worth of discussion if the realization of the prototypical implementation can be seen as part of the implementation phase. The results of the following evaluation of the implemented mechanisms can be negative. The prototype should not be used in productive environments. We suggest classifying this as part of the implementation, because prototyping is recommended just for a part of the overall system, namely the coordination mechanisms, itself. Aspects like the adaptation of the planning system or the definition of conversational policies and behaviors should only be implemented ones. Moreover, by suggesting an iterative process execution model, an unsuccessfully evaluated one of the coordination mechanisms leads to a jump back to an earlier phase of the ECo process, namely the selection phase, and then another implementation and evaluation phase is going to be performed. After appropriate coordination mechanisms

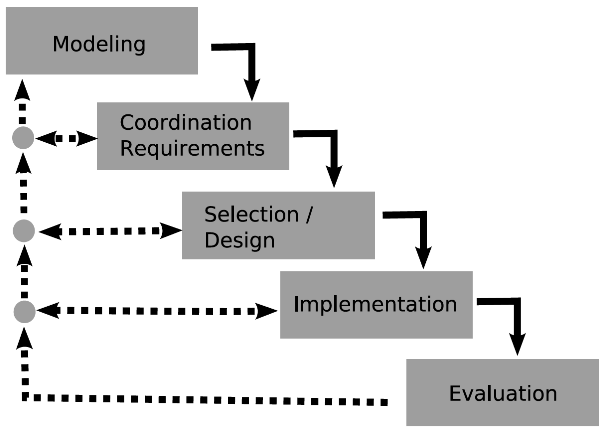


Figure 4.1.: ECo process model

have been identified, also positively evaluated in the quantitative evaluation, it becomes necessary to replace the prototypical implementation in favor for a more efficient implementation of the selected coordination mechanism. In the following each process step is detailed.

4.2.1. Modeling the scenario

Modeling is a necessary step towards the requirement elicitation. A major task while building a model of the scenario is to develop a domain understanding and to identify existing dependencies within the model and to model them explicit, e.g., by means of a dependency graph. Among others the model has to fulfill a number of goals:

- During the modeling of the domain the dependencies between the planning systems are located more precisely. Moreover, the data and data sources that are available/required for the coordination have to be identified.
- The description of the requirements base on the terms and concepts introduced in the model. Thus, the model defines the vocabulary for the following steps of the ECo process.
- Based on the model the evaluation criteria have to be defined, as well. Therefore, the model provides the terminology. During the modeling process suitable scenario specific evaluation criteria can also be identified.

Based on these goals a formal modeling of the environment is reasonable. A formal model of the scenario enables the definition of coordination requirements and evaluation criteria in a formal way, avoiding ambiguousness and even enables a formal validation of these criteria.

For formal modeling of domains and especially requirements von Lamsweerde [Lam09, Chap. 4.4] discuss the following techniques:

- specifications based on temporal logics,
- state-based specifications, using Z for instance,
- event-based specifications, e.g., by using Petri-Nets or statemate, and
- algebraic specification.

We have already presented a common formalization of planning problems in Section 2.2.1 and the modeling of dependencies in the TÆMS modeling framework in Section 3.1.2. Both approaches use algebraic descriptions of their problem space. Note that TÆMS can be visualized as a graph but is based on an algebraic definition (see [Dec95, Chap. 3] for details). Most planning problems are modeled based on an algebraic specification. This indicates that this form of modeling is well suited for modeling planning problems and therefore might be well suited for the modeling of coordination problems among multiple planning systems, too.

At this point we do not suggest a specific modeling language as the field of potential application is broad and the design of an appropriate general modeling language is not intended in this study.

Independent of the chosen modeling approach, it is important to keep in mind that the purpose of the modeling is to define the terms required to define the coordination requirements and evaluation criteria that are used to validate the selected mechanisms. Thus, a formally grounded model is favored.

Example

To give a more concrete impression of modeling a coordination scenario we present it by an example. Therefore, we use the example of the production and distribution of goods, introduced in Section 2.1.1. In this model we do not deal with the internal modeling of each of those sub problems but more on the input/output behavior of the planning system and the overall flow of goods.

As already mentioned, the task of scheduling is to assign operations to resources. An operation o is defined $o = \langle r, \mathsf{d} \rangle$ with

- r, an identifier for the resource this operation has to be performed on
- d, the duration the operation takes on the resource.

Let $\mathcal{O}$ be the set of all operations. Based on the definition of operations we can define a product p as follows: $p = \langle \mathsf{we}, \mathsf{h}, \mathsf{w}, \mathsf{d}, o_1, \dots, o_n \rangle$ with

- we, the weight of a product,
- $\mathsf{h}, \mathsf{w}, \mathsf{d}$, the geometrical information of this product, as a box with height, width and depth.
- $o_1, \dots, o_n$, with $o_i \in \mathcal{O}$ a finite sequence of operations that have to be performed to produce the product.

An order d can be modeled as a quintuple $d = \langle p, \mathsf{q}, c, \mathsf{d}, \mathsf{e} \rangle$. Thereby, the elements of this tuple are defined as follows:

- p, is a product key, specifying the product that has to be produced.
- q, the quantity of the product that has to be produced
- c, is the customer, which encodes the location of a customer by its x and y coordinates.
- d, the due date, at that point in time the order should be finished.
- e, the penalty that has to be paid to the customer per time unit of lateness.

Based on a set of orders $\mathcal{D}$ the production schedule can be generated. The generated schedule consists of a set of assignments. One assignment for each operation that has to be performed. In the subsequent process we are not interested in details how the schedule is formed. What in fact becomes important is the finish time of a good. We refer to a good as an instance of a product that is produced for a specific order. A good (g) is finished if all its operations have been performed. A finished good f can therefore be defined as $f = \langle \mathsf{t}, g \rangle$. That is at time t the good g is finished. The set of all finished goods $\mathcal{F}$ can be extracted from the local production schedule.

For the packing planning system for the finished goods the set $\mathcal{F}$ is planning relevant information. Moreover, information about the *bins*, the loading devices, is necessary. A loading device type has information about its volume and the costs induced per used instance. Additionally, there exist constraints for the packing problem, like the maximal number of loading devices that can be loaded in parallel. The packing, however, has to load the daily production on loading devices to pass them to the transportation unit. As we have now outlined the inputs describing the packing problem, we have to define the output, the packing plan. The result of a packing planning is a list of assignments that assigns each product to exactly one loading device. Again one can abstract the details of the plan only to the relevant information, required for the transportation planning. A ready to ship loading device rl is defined as $rl = \langle \mathsf{rt}, \mathcal{D}_{rl} \rangle$. Thereby, rt is the time of availability of the loading device and $\mathcal{D}_{rl}$ is the set of orders, for which goods are loaded on the device. From this point onwards the loading device can be shipped by a truck. A packed loading device has at least one item for an order of this list loaded.

As pointed out above, the locations of customers are given as coordinates in a Cartesian coordinate system and distances can be computed accordingly. For all trucks it is assumed that they have the same speed. Thus, distance and traveling time have a linear relation. All trucks have the same capacity of loading devices. For each daily production the trucks have to deliver the goods to the customers. The result of the transportation planning is a sequence of customers that are served. For each stop the time of arrival can be computed. If a truck arrives at a customer's side, the loaded amount of goods is removed from the loading device. A customer's order is satisfied if the number of products specified in the order has been delivered to the customer.

4.2.2. Coordination requirements

In the second step of the ECo process *coordination requirements* are introduced. They define requirements a coordination mechanism has to satisfied in order to be applicable for the given problem. The term coordination requirement has been used in the scientific literature, already. But in existing work different notions of this term has been used. For instance, Cataldo et al. [CWHC06] use the term to indicate that software developers should communicate and coordinate their activities in the development of a software product. Thereby, their research focus merely on the organization theory and collaborative working environments in the field of software development processes. A similar usage of this term can be found in Zelewski and Siedentopf [ZS99]. Within the research of DAI the term has also been used by Singh [Sin98]. Thereby, the notion is different in the sense that a coordination requirement is a requirement of a coordination mechanism towards the behavior of its participating agents.

Here, another notion of coordination requirements is used. Coordination requirements are requirements that characterize properties of a coordination problem that have to be satisfied to apply a coordination mechanism in the given problem/context. Thus, coordination requirements specify the requirements towards the coordination mechanisms and consequently guide the selection process of a mechanism. Following the notion of requirements engineering, requirements can be classified as either *functional requirements* or *non-functional requirements* [Lam09, pp. 23,24]. Defining functional requirements in the context of the coordination of autonomous planning entities might seem straight forward; the plans have to be coordinated. But this has to be defined more precisely. To do so we have to distinct different cases of dependent plans, i.e., different ways how the partial plans can be composed and interleaved:

- *parallel execution*: Different entities plan their activities independently and execute them according to their local reasoning. This situation can be found, for instance, in classical AI planning, where different action planners formulate a plan that has to be coordinated, e.g., by plan merging techniques (see Section 3.1.2). Two plans are executable if no conflicts occur in all possible interleaving points of the plans. Details have been discussed in previous sections.

- *concurrent execution*: In contrast to parallel execution each planning system computes a schedule. The execution of each action in each plan starts at a given point in time. These schedules are executed concurrently. Such a situation could be found, for instance, in the supply chain management scenario (Section 2.1.2). Here primarily precedence constraints and, if applicable, constraints concerning shared resources are of particular interest during the coordination. Consequently, two plans are concurrently executable if they do not require the same resource at each point in time and all precedence constraints between the objects that are scheduled, e.g., operations to fulfill an order, are satisfied.

- *sequential execution*: Here the execution of the partial plans is not done at the same time. Consider the production and distribution of goods case study (introduced in Section 2.1.1), for example. All goods have to be produced before they can be packed and shipped. The material flow is strictly sequential. Thus, the planing step for dealing with a specific good is sequential as well. Coordination has to ensure that this sequential relationship is guaranteed in the different plans. Consequently, two plans are sequentially executable if all precedence constraints are satisfied between the objects of the plan and all plans are complete, i.e., providing all necessary actions for all goods.

- *compositional execution*: If different planning entities generate parts of an overall schedule that can only be executed if all parts work together in a coordinated manner, i.e., no plan base on incorrect assumptions concerning the other plans. An example of such a coordination problem is the container terminal management problem introduced in Section 1.2. One example in this scenario is that the berth allocation solver assumes a processing time of the ships that depends on the number of assigned quay cranes that is determined by another planning system.

The definition for dependency among plans depends on their characteristic joined execution.

Beside this functional requirements a number of non-functional requirements might have to be defined in different scenarios that constraint the coordination mechanism in the way how it achieves coordinated plans [Lam09, p. 24]. These non-functional requirements are specific for each scenario, and are essential for the identification/selection of a coordination mechanism, as they encode characteristics that the coordination mechanism has to have, to be applicable in the scenario. Therefore, it is essential that these coordination requirements are identified properly. This is done applying an elicitation process. This process is motivated by the requirements engineering process. The elicitation process suggested by von Lamsweerde [Lam09, p. 34] has four phases that are executed iteratively in a spiral model. The phases are:

- domain understanding and elicitation,
- evaluation and negotiation of the requirements with the stakeholder,
- specification and documentation of the requirements, and
- quality assurance.

The domain understanding phase corresponds to the modeling discussed in the previous section. The elicitation of the requirements depends strongly on the domain a coordination requirement is required for. If a coordination scenario is designed, it is important to discuss these issues with the stakeholders, i.e., the entities that have to coordinate their plans.

The evaluation and negotiation of requirements is an important aspect, as some requirements are show-stoppers, i.e., they have to be satisfied totally to ensure applicability, e.g., to ensure privacy in form of information hiding in the coordination of the plans of different companies in a supply chain. Other requirements might be optional or should be optimized up to a degree that is economically reasonable, e.g., needed computation time. For those weaker requirements it may make sense to define acceptable thresholds that should be met to make them strict requirements, as well, e.g., it might be unacceptable to wait for a coordination of next week production for longer than two days.

As already mentioned, in the section concerning modeling, it is important to specify the coordination requirements in a formal way that enables a strict distinction between satisfied and non-satisfied requirements of a coordination mechanism.

The specified requirements should be checked if they are consistent and should be validated with the stakeholders, to ensure that the coordination requirements are specified correctly and adequate specifying the needs of the stakeholders.

These activities are summarized as quality assurance. This underlines the notion of coordination requirements that should guide the selection process according to the needs of the current situation at hand.

Example

Again, we want to clarify the definition of coordination requirements using an example. Suppose we want to define the coordination requirements for the production and distribution of goods case study, that has been modeled in the previous section. As pointed out above, in this scenario the plans are executed sequentially, at least from the point of a planning object, i.e., a good that has to be scheduled first, then packed and finally transported. It is necessary that the plans are consistent, i.e., feasible if all precedence constraints are satisfied. The precedence constraint in this scenario is defined in a way that a good cannot be packed before it has been produced. Therefore, the input for the packaging planning is the set of finished goods $\mathcal{F}$. This set has to be consistent and complete. $\mathcal{F}$ is complete if all goods necessary to fulfill the orders in $\mathcal{D}$ are contained (see Equation 4.1).

$$\forall f \in \mathcal{F} : good(f) \in \mathcal{G} \tag{4.1}$$

$\mathcal{F}$ is consistent if for each $f \in \mathcal{F}$ the last production activity has been finished before it can be packed. This aspect is covered by the definition of f as it guarantees that the condition referred to in Equation 4.2.

$$ttime(f) = \max_{\forall a \in \mathcal{A}g} end(a) \tag{4.2}$$

For more details we refer to the evaluation section of this case study (Section 5).

4.2.3. Selection of appropriate coordination mechanisms

After the domain has been described and requirements have been defined the next step of the ECo process is the selection of appropriate coordination mechanisms. This is done applying a *qualitative evaluation*. The goal of the qualitative evaluation is to prove, or at least give strong indications what coordination mechanisms are well suited for the given situation, to fulfill the coordination requirements. There are at least three ways the conformance of a coordination mechanism can be validated:

- *by reuse*, i.e., for standard requirements, the result is known and can be looked-up in a repository. This is clearly the most convenient way. But it has two drawbacks: First, such repositories do not exist so far in an appropriate way. Within this thesis the foundations for building such a repository are laid out. Second, this approach should only be used for requirements that are not specific for a scenario. A fairly simple example of such an repository is shown in Table 4.2 (page 118).

- *by design*, i.e., some mechanisms satisfy some requirements by their design, either intentionally or unintentionally. An example are auctions that do only provide information about a monetary validation of a certain good, but no motivations for the value, the true local value, or how each participant can fulfill the task he is bidding for. By that design auctions are, for instance, a possible option if information hiding becomes an issue as a coordination requirement. Another example is the usage of an interface of an existing planning system. If the coordination mechanism can interact only through an interface with the planning systems that specify a simple set of interactions, like defining the planning task and reading the results, the coordination mechanism gets no insides of the planning system. Thus, the planning system is seen as a black box to the coordination mechanism, which can be a coordination requirement if existing planning systems have to stay in use or a modular system architecture is desired.

- *by proof*, i.e., this is typically the hardest form of validating the compliance of a coordination mechanism with a coordination requirement. A formal proof has to be designed that shows that a formalized requirement is satisfied during the coordination process. This can be done, e.g., for the amount of information published during a negotiation. If a metric exists for the amount of information that is allowed to be published, it can be verified or falsified that the coordination mechanism can hold the information hiding requirement. Another example might be the avoidance of inconsistent states at different planning agents, e.g., by model checking techniques. These proofs, of course, are only possible if the agent's local state can be accessed centrally. But as this step is done before the roll-out for analysis this can be assumed to be given.

Example

We take an abstract point of view here and make a selection on the base of the more general characteristics presented in Section 4.1 and the classification summarized in Table 4.1 (page 116) in particular.

For the scheduling and distribution of goods case study, the problem was characterized as no allocation problem, no comparable objectives, heterogeneous planning problems, the existence of an overall objective, no need for information hiding and no cyclic dependencies. Consequently, the following mechanisms can be filtered for a more detailed consideration:

- plan merging,
- decentralized planning for a centralized plan,
- result sharing, and
- negotiation.

The supply chain management case study was characterized by an allocation problem, possible comparable objectives (money), possible heterogeneous planning systems, an overall objective function (serving the customer on time), the need for information hiding among the partners of the supply chain and no cyclic dependencies. According to this classification the following coordination mechanisms can be investigated in more detail:

- mediator-based coordination,
- auctions, and
- negotiation.

The container terminal management problem is characterized by no allocation problem, no comparable objective functions, heterogeneous planning problems, no overall objective function, no need for information hiding and the existence of cyclic dependencies. Therefore, only negotiations seem a reasonable approach.

4.2.4. Implementation of coordination mechanisms

After appropriate coordination mechanisms have been identified they have to be implemented. To ensure an efficient implementation of the coordination mechanisms, we propose the CoPS (Coordination of Planning Systems) process that offers support for a number of necessary steps that have to be done. Complementing the CoPS framework offers a software framework that facilitates the implementation of coordination mechanisms more quickly, by providing an infrastructure. Note that the CoPS framework complements the CoPS process and is not mandatory, if the CoPS process is used. Therefore, the framework has more flexibility to evolve or can be replaced entirely for a more customized solution for a given situation. The CoPS process and framework are described in detail in the next section. Therefore, we give only a brief overview here over the necessary steps of the CoPS process which are necessary to perform. The activities are on the level of the entire network of planning entities that want to coordinate their activities, as well as, activities that have to be performed within each of those planning entities. The first will be referred to as the global layer, while the latter ones will be referred to as the local level.

CoPS process on the global level

The entire selection process of coordination mechanisms is performed on the global level, as the coordination mechanisms have to be defined for the entire network of planning entities, and all these entities have to act accordingly to the selected coordination mechanisms. During the modeling of the domain the effects that cause the need for coordination should have been identified. Thus, it is clear in which situations coordination becomes necessary. For these situations coordination mechanisms have been selected in the previous step, according to the defined coordination requirements. Note that for different situations the need for coordination occurs, possibly different coordination mechanisms have been identified, or assumed to be beneficial. We are focussing here only on coordination based on direct communication. So the coordination mechanisms are guided by an interaction protocol. Consequently, these interaction protocols have to be defined for the selected coordination mechanisms. Thereby, it is strongly advocated to use, if possible, existing interaction protocols, as they have proven useful. But this is, of course, not mandatory. Thus, for the implementation phase of the coordination mechanisms appropriate interaction protocols have to be defined that implement each coordination mechanism on the global level.

Moreover, some infrastructure, like a centralized coordination agent, has to be set up, if necessary. The CoPS process is detailed in Section 4.3.

CoPS process on the local level

On the local level, i.e., within each planning entity, a number of adaptation steps have to be performed. First of all, the local planning/scheduling systems have to be made *accessible* for the coordination mechanisms. As pointed out before, each planning entity is represented by an agent, called planning authority agent. Such an agent implementation is provided within the CoPS framework. This agent must have means to access the local planning systems in a way that it is possible for the agent to specify a planning problem and to get a solution for this problem.

The next task is to define local strategies how the agent should behave during the coordination process. The basis for this strategy definition are the defined interaction protocols given by the global level and the selected coordination mechanism that is currently implemented. If this coordination mechanism implies a dominating strategy (see Section 3.1.3) for the planning entities, this strategy can also be given by the global layer, otherwise a local strategy has to be defined. This strategy has to be defined by human managers, and has to be implemented in a way that the agent can decide how to behave within a certain conversation with other agents coordinating its activities. Therefore, the strategy has to be broken into conversation behaviors that specify for each point in the conversation process; guided by the interaction protocol, what answer should be sent, if any. These behaviors depend on a number of information that is specific for each planning entity and has to be defined in an appropriate way for the planning entity.

It is also necessary to encode domain knowledge, e.g., how to react efficiently in case of dynamic changes or during the search for local plan improvements. Procedures to achieve those effects are often known by local experts and can improve the system's performance and efficient use of the local planning system.

As mentioned above, the CoPS framework is designed to facilitate a fast prototypically implementation of a coordination mechanism and is therefore primarily designed to be easily adoptable and not necessarily towards an efficient realization of a special coordination mechanism. Therefore, it is advocated to use the CoPS framework for the evaluation of possible candidate mechanisms and for the usage in a pilot. Before a roll-out in a production environment it may be therefore useful to adapt the framework or use a special implementation facilitating the selected coordination mechanisms. More details concerning the CoPS process and the CoPS framework are given in the next Section 4.3.

4.2.5. Evaluation of coordination mechanisms

The task of the evaluation is to investigate if the desired performance criteria are met by the implemented coordination mechanisms and to identify the most suitable mechanism. This evaluation is named here *quantitative evaluation* of coordination mechanisms. Thereby, it strongly depends on the given situation which performance measurements are useful. Measures can range from an overall social welfare metric to more subjective criteria. Additional performance measurements, like the needed time for reaching a coordinated solution, can be measured and can be relevant for the practical usage of a coordination mechanism, as well. Even though most of the evaluation criteria are scenario specific and should have been defined in the domain modeling phase of the ECo process some concerns regarding the quantitative evaluation can be discussed here. Suppose we are interested in measuring the quality of the coordinated plans, i.e., the result of the coordination mechanism. Assume that this result can be expressed as real value. In the following three different types of analysis are discussed that can be applied in different scenarios, to investigate the quality of a coordination mechanism in quantitative terms. These evaluations are:

- baseline comparison,

- comparison with other approaches, and
- c-competitive analysis.

The first type is a *comparison to a baseline*. It is probably the most simplistic kind of comparison. It is mainly characterized by comparing the situation before the coordination with the result after the coordination. Thus, the leverage of the coordination mechanisms can be measured. Thereby, problems concerning the evaluation occur if the plans before coordination are not feasible, i.e., they could not be executed without generating conflicts. If the plan is infeasible after the coordination as well, one could measure the violation of such hard constraints, but for sure this will not be a desired outcome and the quality of a coordination mechanisms is not evident if the plans are not coordinated in at least a feasible way after the coordination. If the coordinated plans are feasible and before were not feasible how can the improvement be quantified? If infeasible situations are penalized in the objective function, the improvement could be measured and will be significantly high, even though the feasible solution might have a low performance in contrast to a potentially optimal set of coordinated plans. If this is not possible, one could use a sorted set of quality metrics. So, one quality metric is feasibility, and a feasible plan, of whatever quality, dominates a non-feasible plan, if it is rated high by an objective function. In a situation where a set of uncoordinated infeasible plans can be transformed into a coordinated feasible set of plans the improvement cannot be quantified. Instead it can be measured that the plans are on a higher quality level. The advantage is that no continuum of good infeasible plans and bad performing feasible plans exists that can be mixed up.

If different coordination mechanisms have been implemented, a *comparison to other approaches* can be achieved. In contrast to the comparison to a baseline, this approach gives an impression of the gained quality that has been achieved. Nevertheless, it requires different coordination mechanisms to be implemented for the given situation. This is supported by frameworks, like the proposed CoPS framework, but in any case it results in an extra effort for designing and implementing different coordination mechanisms. There exist two special cases of this comparison. The first one is the usage of testbeds. A testbed is a standardized platform for experiments. Testbeds are widely accepted in the field of DAI, see Decker [Dec96] for an early survey. Testbeds offer the functionality to represent different instances of a fixed application problem and facilitate to implement different solution mechanisms for these problems. In contrast to most frameworks, testbeds are designed specifically for one type of problem. They typically offer an interface for the decision making process, to evaluate different problem solving strategies for the specific problem class. Often testbeds come along with fixed scenarios for benchmarking or generator for problem instances. Testbeds have been proposed for different fields of application, like distributed vehicle monitoring, presented by Decker [Dec96], manufacturing control, presented by Cavalieri et al. and Verstraete et al. [CBM$^+$99, VVB$^+$06], or market-based negotiation systems[6] presented by Collins et al. [CTMG98]. If testbeds for the given problem at hand exist, this is an ideal way to gather information about existing mechanisms or to compare a generated solution with existing ones. Of course, therefore the local problem has to be transformed into a format that can be represented in the testbed if applicable. As testbeds aim at a general class of problem types, special instances might not be covered entirely.

Another way to evaluate an implemented solution is to compare it with the optimal solution. Typically, the computational effort to find the optimal solutions is very high, and can only be done for small instances. A similar form of comparison is to use a centralized computed solution, which typically outperforms the coordinated plans, if the centralized planner has all required information that is available, and no time limits for the centralized planner exist. Such a benchmark is presented, e.g., by Wörner and Wörn [WW06b]. Note that results of such a benchmark are not useful in the selection process of a coordination mechanism. If a comparison should be used in a selection process, a “fair” comparison is required. The notion of fairness is discussed below.

The *c-competitive analysis* stems from the field of online algorithms. An insightful introduction into this field is given by Borodin and El-Yaniv [BEY98]. Its goal is to evaluate the quality of an online algorithm. This analysis is well-suited to evaluate the capability of mechanisms that have to deal with dynamics. In a c-competitive analysis the maximal factor is computed that expresses to what degree the online algorithm is worse in comparison to the optimal solution. Thus, the performance of both solutions is computed. And the different performance values are analyzed to identify the largest performance gap. This gap is used to compute the *c-factor* that expresses to what degree the online algorithm is worse off in comparison to the optimal solution.

Of course, this technique requires that the optimal solution is known. If this is not true, or the computation of the optimal solution becomes intractable, a modified comparison technique is necessary. In the following, we call the comparison with a known optimal solution a *strong* c-competitive analysis. In contrast to a *weak* c-competitive analysis, where the result of the online algorithm is compared to the best-known solution by an offline mechanism. An instance of such a weak c-competitive analysis has been performed, e.g., by Timmermann and Schumann

[6] `http://magnet.cs.umn.edu/index.php` accessible 03/05/2010

[TS08].

If results of different mechanisms are compared, the fairness of this comparison becomes an issue. Fairness thereby has at least two aspects that are discussed here. First of all, the functionalities of the mechanisms have to be comparable. A typical example of an unfair comparison that violates this aspect is, for instance, the benchmark/comparison of a MAS that is capable of reacting to dynamics and a centralized planning system, e.g., based on operations research techniques or classical planning and scheduling techniques, without capabilities of reacting to changes and adapting its plan. If those two systems are compared in a dynamic environment, the MAS outperforms the centralized planning system easily. Such a comparison does not allow to make a statement concerning the abilities of multi-agent systems or centralized planning systems, except that one system can handle dynamics and the other one cannot. This does not allow any judgments concerning centralized planning and scheduling systems in general, because those approaches can also be equipped with techniques for handling dynamics, as already discussed in Section 2.2.3.

Another type of "unfair" comparison can be found if two mechanisms are compared that have to respect different sets of restrictions. For example, two approaches satisfy different sets of coordination requirements, or other constraints are opposed to the systems. Typically, less restricted systems can find solutions of higher quality. By opposing more restrictions, the change of finding high-quality solutions is decreased. Consequently, the comparison with a mechanism that does not obey to the same constraints will result in an unfair comparison, like the one sketched in Figure 4.2.

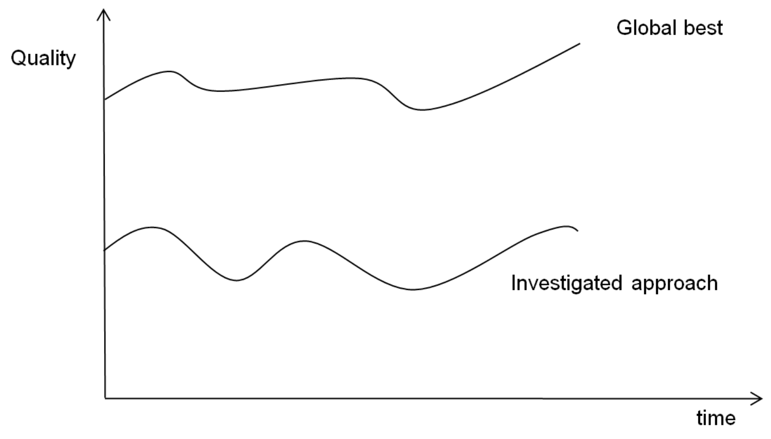


Figure 4.2.: Example of an unfair comparison

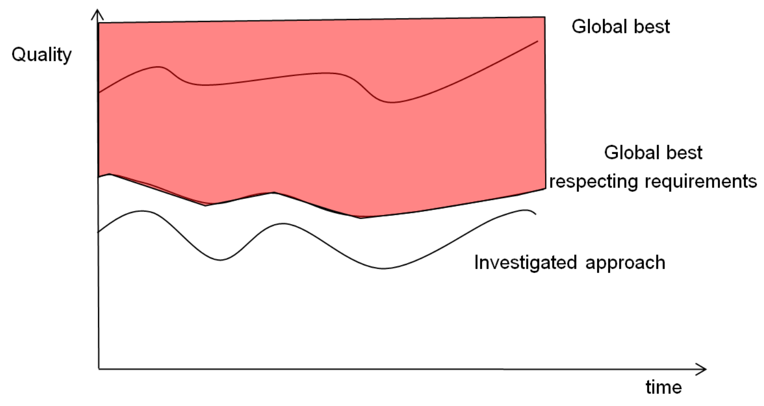


Figure 4.3.: Example of an fair comparison

A fair comparison would instead compare the results of the mechanism under investigation with the results of the best possible results of a mechanism that respects all coordination requirements. This is sketched in Figure 4.3. A fair comparison gives a more realistic impression of the capabilities of the mechanism under investigation.

Nevertheless, an unfair comparison can offer additional information, of course. For instance, comparing the result of coordination mechanisms that respects all defined coordination requirements with a solution computing the global optimal solution and thereby violating a number of coordination requirements enable, the estimation of the *costs* for solving this problem in a distributed way, and the *costs* of the coordination requirements, which can be the starting point of a discussion concerning the coordination requirements, i.e., is it rational to keep these requirements, or can they weakened, to ensure a better overall performance. This issue concerns the tactical and strategic decision making, and the field of how problems should be distributed, and therefore is not in the scope of this study, but remaining an important issue for future work.

4.3. CoPS process and framework

The CoPS process and the CoPS framework have been developed to support the implementation of coordination mechanisms as part of the ECo process. The CoPS process and framework support the ECo process, as they detail out one specific step of the ECo process. Nevertheless, it is possible to use the ECo process and the

CoPS process and framework independently. The ECo process does not demand the usage of the CoPS process or CoPS framework, and vice versa. Each planning entity is represented by its agent. This agent is called the *planning authority agent* (PAA), because it represents the planning authority, which is a synonym for planning entity.

Coordination mechanisms among agents are used to coordinate the local plans. To implement these mechanisms, every agent has to respect two aspects that guides its local behavior. First, the agents have to behave accordingly to the protocols defined on the network layer. Second, they also have to behave accordingly to the specifications of the local planning entity that they represent.

The idea of the CoPS framework is that not every planning entity has to design and implement its own agent, but rather instantiate an agent derived from the CoPS framework, that have been adapted to the local situation at hand. This emphasizes the re-use, and more important for the planning entity, a shorter development phase and less development costs. To facilitate the reuse of the CoPS framework, it is important to ensure that the (PAA) behaves strictly as specified by the planning entity and that they can be adapted easily. Within the CoPS process it is described how such agents can be customized for a planning entity. But even if the CoPS framework is not applied to coordinate the behavior of planning entities the CoPS process can offer valuable guidelines, as it addresses issues like adapting agents to existing planning systems, specifying conversation protocols and conversation behaviors, which are topics that are not exclusive for the CoPS framework, and in case of the specification of conversation protocols and behaviors even not specific for the coordination of planning systems, at all.

4.3.1. Concept of the coordination process in the CoPS approach

We use agents to represent the planning entities that have to coordinate their activities. Moreover, a coordination agent (CA) is assumed to be in place that enables some centralized book-keeping and offers services like indirect communication to the network, if necessary, e.g., for information hiding issues. But more important this coordination agent represents the network as a whole. A CoPS network is setup by specifying a coordination agent. Other PAAs can register themselves at the CA to become part of the network. In this thesis only the operational coordination of activities is addressed. Thus, it is not in the scope why and when an agent enters a network. We assume that the agent is ordered to enter a specific network by its planning entity. Therefore, it is reasonable to assume that it has been prepared to do so, e.g., by equipping the agent with the shared ontology of the specific network and conversation policies and behaviors for that network. To foster a faster implementation of coordination mechanisms, the CoPS frame-

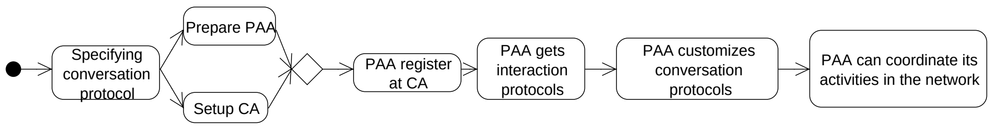

Specifying conversation protocol
Prepare PAA
Setup CA
PAA register at CA
PAA gets interaction protocols
PAA customizes conversation protocols
PAA can coordinate its activities in the network

Figure 4.4.: Process of setting up a network, according to the CoPS process

work offers a way to reuse the implementation of conversation protocols. These protocols have to be specified on the global network level. Code implementing these protocols is generated and is distributed among agents, when they register to participate in the network. Thus, it is not necessary that every planning entity has to implement the conversation protocol. It can focus on the aspects that are specific for its own local context. These protocols can be enriched by conversation behaviors locally defined. After this has been done the agent acts according to the conversation protocols defined for this network. Therefore, the CA functions as an embodiment of the organization, i.e., the network, it represents. The embodiment of organization is also favored by Piunti et al. [PRBH09]. The process described above is depicted in Figure 4.4 using an UML activity diagram notation. For an introduction into UML we refer to Oestereich [Oes09, Chap. 4.6]. More technical details of the CoPS framework are discussed in Section 4.3.3.

4.3.2. CoPS process

The CoPS process can be used to take the necessary steps to implement a coordination mechanism. It is assumed that the selection process of the ECo process has been performed and one or more mechanisms have been selected for implementation. The CoPS process supports the implementation of coordination mechanisms. While the ECo process is executed on a network wide level, i.e., the decisions are made for the entire network. The CoPS process is executed on both levels, the global network wide level and in large parts it comprises process steps that have to be executed on the local layer, within each and every planning entity. Primary it describes how to adapt a PAA of a planning authority to the coordination mechanism that has been agreed upon on the global level. An overview of the CoPS process is given in Figure 4.5.

The CoPS process defines five steps. The specification of conversation protocols is given on the one hand in form of an interaction protocol but also offers additional information about the expected behavior of the agents. This is necessary as interaction protocols are not sufficient for specifying agent interactions (see Bartolini et al. [BPJ03]). This process step is done on the network wide

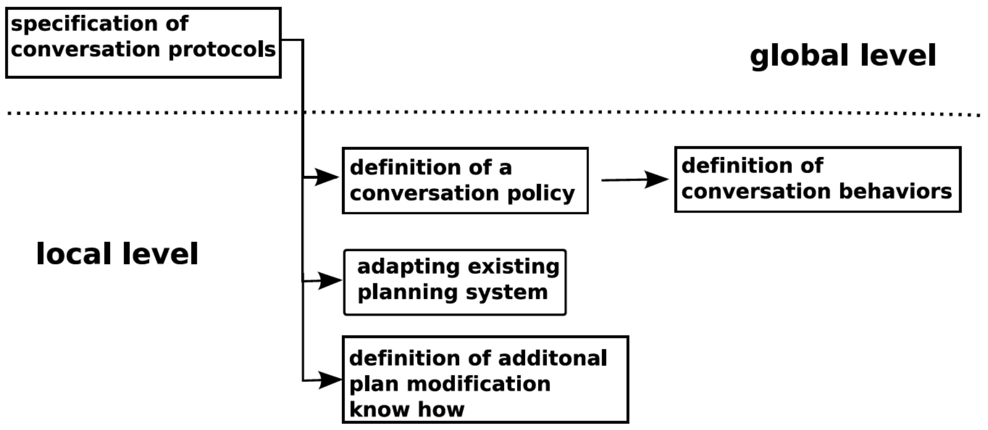

specification of conversation protocols
global level
local level
definition of a conversation policy
definition of conversation behaviors
adapting existing planning system
definition of additonal plan modification know how

Figure 4.5.: Overview of the CoPS process

global layer. In contrast to the previous steps on the global layer, it is a purely technical step. The required conversation processes for the selected coordination mechanisms have to be modeled.

If the conversation protocol has been defined, each planning entity has to define its conversation policy. This policy compromises guidelines like its willingness to make concession and to whom it is willing to do so, or its willingness to publish which information to whom. This conversation policy is fundamental for the next step, the definition of the conversation behaviors. A conversation behavior fixes for each and every step during a conversation, which action, i.e., what kind of answer, the PAA should provide.

Doing so the PAA probably requires additional information, and to coordinate the plan of its planning entity, it has to have access to the local planning system. Therefore, the planning system has to be adapted to be accessible/usable for the PAA. Finally, it is necessary for the PAA to guide the search for a coordinated local plan to explore the consequences of modifications of the inputs of the planning system. Therefore, the PAA has to have means to modify the inputs in an informed way and not only by uninformed search. The search process can be shortened, if knowledge of the local search space is incorporated within the process. Therefore, it is useful to encapsulate this knowledge in a separate module that agents can use. Thus, the domain knowledge can be separated in a module that is not part of the CoPS framework. Therefore, the core implementation of the agent, taken from the CoPS framework does not have any domain knowledge. The agent can make use of it by the localized module.

Specification of conversations

A conversation is characterized by an interaction among two or more entities, acting on two roles. Even if the number of participants is larger the interaction can be described by the behavior of two roles. This concept is, for instance, used in the definition of the contract net protocol or auctions protocols, as well.

Conversations are specified in three parts. An overall interaction protocol, a conversation policy, and conversation behaviors. An interaction protocol specifies correct sequences of messages exchanged in a conversation. A conversation policy encodes behavioral guidelines for all conversation behaviors. It encodes additional information that is not covered in the interaction protocol. Conversation behaviors specify the behavior an agent performs after it has received a message as part of a conversation. We assume that conversation behaviors are defined by humans. Therefore, it is not necessary that conversation policies have to be machine processable. However, it would facilitate the construction of more advanced agents, that can interpret these rules and behave accordingly. Moreover, a formal notation disambiguates the notation of conversations.

Specifying interaction protocols First, we discuss the specification of interaction protocols. These protocols have to be readable by humans: during the design process and the subsequent steps of the CoPS process. As outlined before, it is also advantageous to specify interaction protocols in a machine readable format to enable an automated generation of state-based implementations for both roles of the conversation.

Therefore, we suggest using a textual notation to describe these protocols. Based on this representation it is possible to generate a sequence diagram, allowing easy human readability. Moreover, the textual representation is parsable to generate the necessary automata for both roles. Using a textual representation for the description of agents interaction specification has been proposed by Koning and Romero-Hernandez [KRH03], as well. In their paper they use a textual representation of AUML diagrams to generate Promela code that can be verified by the model checker Spin to verify certain aspects of the specification of the interaction. The authors do not use the textual representation as a mean to foster implementation and communication towards the developer.

We have surveyed textual notions for sequence diagrams that can be used to generate graphical representations. Tools like the WebSequenceDiagrams[7] or Quick Sequence Diagram Editor[8] offer to specify UML sequence diagrams in a textual form. Keywords, e.g., for defining alternatives have been defined in the UML stan-

[7]`http://www.websequencediagrams.com/`, Accessed: 03/09/2010
[8]`http://sdedit.sourceforge.net/`, Accessed: 03/09/2010

dard. Those keywords have been summarized, for instance, by Oestereich [Oes09, p. 364]. Unfortunately, the syntax of the representation these tools use, is not identical and standardized. For instance, drawing a line from object `Alice` to object `Bob` is encoded as `Alice->Bob:Label of the arc` for WebSequenceDiagrams and as `Alice:Bob.Label of arc` for the Quick Sequence Diagram Editor. We choose the notation of the WebSequenceDiagrams tool, because it facilitate the construction of *compilers* that can parse the description and generate behavior automata. A behavior automaton describes the correct sequences of messages from the perspective of one role of the conversation. An example of a textual representation is shown in Listing 4.1.

Listing 4.1 Textual description of the contract net protocol

```
1  participant Initiator
2  participant Participant
3  note left of Initiator: onetomany
4  note right of Participant: bilateral
5
6  Initiator -> Participant : cfp
7  alt
8   Participant -> Initiator : propose
9   note left of Initiator: sync
10  alt
11    Initiator -> Participant : accept-proposal
12    alt
13     Participant -> Initiator : failure
14    else
15     Participant -> Initiator : inform
16    end
17  else
18    Initiator -> Participant : reject-proposal
19  end
20 else
21   Participant -> Initiator: refuse
22 end
```

In the lines one and two the participating roles are defined. Thereby, `participant` is a defined keyword of the WebSequenceDiagrams syntax. The declaration is optional for the generation of the diagram, but assumed to be mandatory for the generation of automata, because it eases the parsing of the text. In lines three and four, two comments (indicated by the WebSequenceDiagrams keyword `note`) are used, to indicate the number of roles that are acting on these roles. The cardinalities of common interaction protocols have been presented in Section 2.3.4 (in particular in Table 2.6 on page 52). We identified two possible situations that can be encountered from the perspective of each role. These are the one-to-many (`onetomany`) perspective, and the bilateral (`bilateral`) perspective. In a one-to-many perspective an agent, acting on a role, has a conversation with a number of agents, while from a bilateral perspective an agent has a conversation with only one other agent. In the example of the contract net, the conversation is a one-to-many conversation from the perspective of the manager role, while the same conversation is a bilateral for all other participants. By the keyword `alt` a conditional branch is declared, that can have alternatives, which are indicated by the keyword `else`. It is also necessary to synchronize a conversation at certain points in time. For example, in the contract net, the initiator (manager) has to wait for proposals and is not allowed to proceed after the first proposal has arrived. This is indicated by the keyword `sync` in line nine. It is used in the comment syntax of the WebSequenceDiagrams tool, the effected role is mentioned, that has to synchronize its behavior, here, wait for responses. The resulting sequence diagram is shown in Figure 4.6[9].

The code shown in Listing 4.1 can be compiled into behavioral automata. This can be done in a single pass parsing. A compiler has been developed for the implementation of this thesis. It takes an input in form of a text file and generates the representation of two behavior automata, one for each role in the conversation. The automata are non-finite variants of a Mealy machine, e.g., presented by Hopcroft and Ullman [HU79, p. 43] with ϵ-moves. The output of the automata can be computed by its current state and its input. Arcs are labeled with input and output. This is done in the following notation `input/output`. An ϵ-move indicates that no input is necessary to trigger a transition. Even though, it might be possible to transform this state machine into a non-finite automata (NFA) without ϵ-moves in a second parsing step, this is not necessary, as it has been stated that NFA with and without ϵ-moves are equivalent, see Hopcroft and Ullman [HU79, p. 26]. Graphical representations of the generated automata from Listing 4.1 are shown in Figures 4.7 and 4.8.

To enable the specification of all kinds of interaction specified by the FIPA, an additional construct for loops is necessary, e.g., to represent the iterated contract net protocol. This notation discussed above has been extended to model those protocols, as well. This extension is discussed in Appendix A using the example of the iterated contract net protocol [FIP02g].

[9]The resulting AUML diagram of the contract net protocol has been shown in Figure 3.8 on page 102.

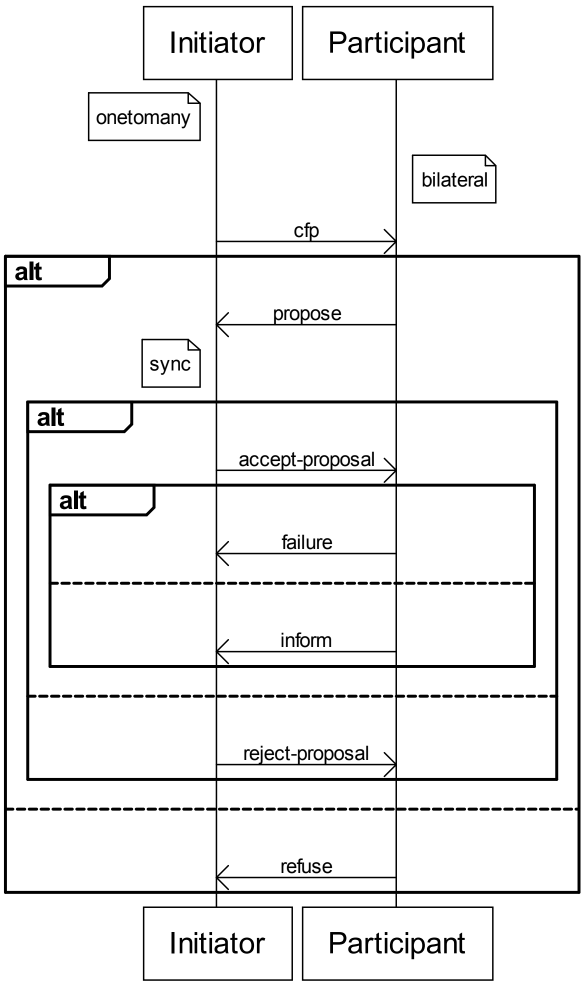


Figure 4.6.: Generated sequence diagram of the contract net protocol

Specification of the conversation policy

A conversation policy is defined by a planning entity to specify general rules that should be obeyed by the PAA in conversations. As pointed out by Kagal and Finin, a conversation policies are “restrictions on communication based on the content of the communicative act” [KF04, p. 121]. This can strive regulations what information is revealed and to whom or what services and service conditions are offered in a specific network, for instance.

Conversation policies have to be described on the one hand in natural language or other concepts, e.g., graphical notations. This kind of notation serves as a tool for communicating the conversation policy to developers or technical staff. These

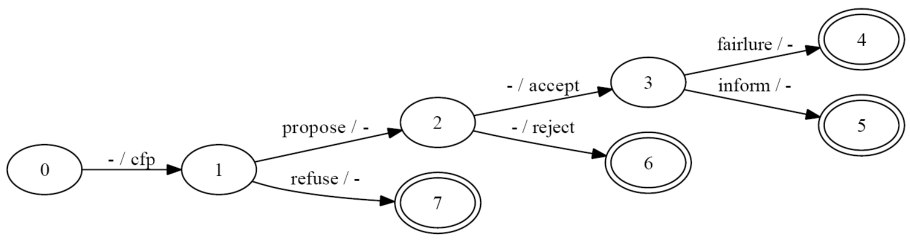


Figure 4.7.: Behavior automaton for the initiator role

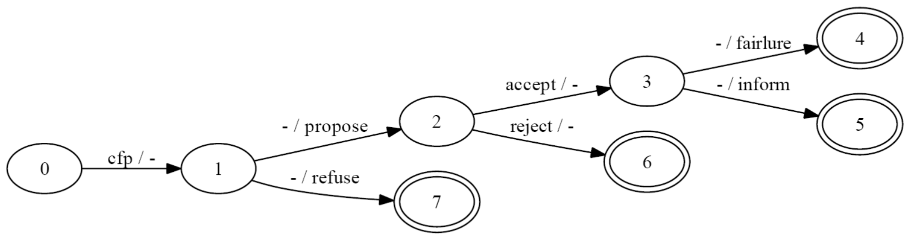


Figure 4.8.: Behavior automaton for the participant role

are the people who are going to implement the conversation behaviors accordingly to the defined coordination policy. On the other hand, the specification should be formal to enable inferring agents to reason about their behavior and validate if it is correct according to their conversation policies. The automated validation can gain confidence in the agent as a representative of the planning entity, because the conversation policy has to be defined in close cooperation with the responsible management. In contrast, the implementation of the conversation behavior is typically done by technical people, who do not have responsibilities concerning strategic decisions, which is encoded in the conversation policy.

Kagal and Finin [KF04] suggested that a conversation policy can be described as a set of positive and negative permissions and obligations. Thereby, the authors define four different notions of policies that could be defined: permission, prohibition, obligation and dispensation. These four aspects are encoded as deontic logic expressions, which determine allowed or forbidden actions for the agent. To resolve conflicts that can arise during the reasoning of the agent, the authors suggest the usage of priority rules. By using such a rule-set it is possible to express what information is revealed and to whom, for instance. The authors mention that the interaction protocols themselves can be expressed with such rules, as well. This is an interesting approach for reasoning agents. But for the design of interaction protocols the usage of rule sets is not well suited for developers. The usage of graphical notations, like the sequence charts, discussed above, are widely accepted and developers are familiar using them. Kagal and Finin use an OWL[10] description to enable automated reasoning about a conversation policy. The usage of this notion of conversation policy enables description and reasoning about aspects of a conversation in more detail, e.g., it is well suited to define what information can be revealed and to whom.

The expression of what actions are allowed and forbidden is a useful step in constraining the actions of an agent. But they are not sufficient for specifying consideration about more value-based concerns, like guidelines how to generate a counteroffer. Therefore, trade-offs are useful to cover this aspect. A trade-off quantifies different valuations between two or more attributes of a conversation object. In the following we discuss the additional value of trade-offs, assuming a conversation to be a negotiation process. Trade-offs enable the detailed specification of the content of messages during a conversation, like a counteroffer or a concession in a negotiation.

Trade-offs have been discussed in the context for multi-attribute negotiations among software agents, e.g., by Huhns and Stephens [HS99, p. 104] or Lou et al. [LJS06].

For the application of trade-offs in automated negotiations it is crucial to acquire the correct trade-off information from the humans, who are kept responsible for the economic consequence of the agent's activities. We have developed a technique to acquire those information, using trade-offs as part of a conversation policy. Therefore, we need a human readable way to specify those trade-offs.

As already mentioned, a trade-off defines how much one attribute can be worsened, in favor for improving another. This can be encoded within a trade-off function. According to Luo et al. [LJS06] this function can be defined as shown in Definition 4.3.1.

[10]OWL is the abbreviation for *web ontology language* see `http://www.w3.org/TR/owl2-overview/` Accessed: 03/11/2010.

Definition 4.3.1

Let the domain of the attribute of the negotiation subject x be defined with $X = [l_x, r_x]$, and let the domain of y be $Y = [l_y, r_y]$. Thereby, we assume, that the domains are sorted in a way that the first value is evaluated at least, and the last value is evaluated highest. Then the function is called trade-off function between X and Y if it is *continuous*, *monotonic* and meets the *boundary condition.* The boundary condition assures, that if one attribute is assigned to the best value, the other attribute has to be made worse [LJS06]. The pair (x, y) is called a *trade-off pair.*

For each trade-off pair a preference function is defined, specifying the preference over the trade-off alternatives. Trade-off alternatives are value combinations of the relevant trade-off pair [LJS06]. Independent attributes are not in a trade-off relation with other attributes. To each of them a preference function is associated.

A trade-off strategy represents a *directed forest*, this is formalized in Definition 4.3.2.

Definition 4.3.2

Given are the negotiation attributes as nodes and the trade-off pairs as directed edges. The direction of an edge is defined by the trade-off function (see Definition 4.3.1). If this function has the form $\tau : X \rightarrow Y$, there exists an edge from node X to Y. Then a *trade-off strategy* represents a *directed forest*:

- Let a be a negotiation attribute. All values of a's value set A are in a preference ordering relation $\preceq \subseteq A \times A$: the preference direction is *descending* if smaller values are allowed, *ascending* otherwise.
- To each trade-off pair (a, b) with A and B as domains for a and b, the following is associated:
  - A trade-off function $\tau : A \rightarrow B$ in terms of Definition 4.3.1.
  - A trade-off preference function $p : A \times B \rightarrow [0, 1]$, which assigns to each trade-off alternative a preference value. It reflects a trapezoid formula of three segments (analogue to the preference function in [LJS06]) to describe the increasing, steady and decreasing preference over trade-off alternatives.
- Independent negotiation attributes are trees with only one node. For each such negotiation attribute a a preference function is associated $p : A \rightarrow [0, 1]$, with A: $\forall a, b \in A : \; a \preceq b \Leftrightarrow p(a) \leq p(b)$.

A resulting set of trees is shown in Figure 4.9. In fact, it is ensured that all trade-

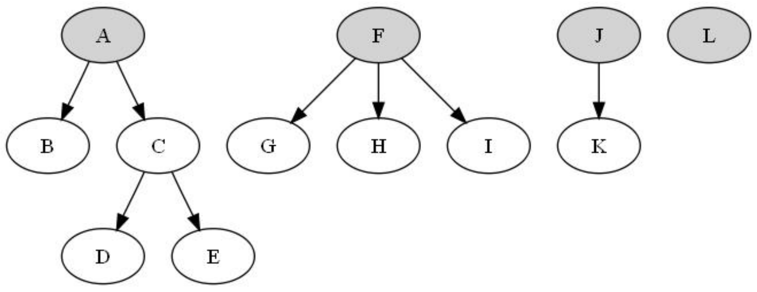


Figure 4.9.: Directed forest representation of a trade-off strategy, [SKT09]

off strategies can be represented as a forest. This ensures formal and informal benefits. A trade-off strategy can be visualized in a *clear* and *accustomed* way to the users. Due to the acyclic structure, the strategy remains consistent. This reduces the complexity specifying and validating those strategies.

We used the Ecore meta model of the Eclipse Modeling Framework[11] (EMF). Details of this framework are provided by Steinberg et al. [SBPM09, Chap. 5], for instance. We use the EMF to define a model for trade-off strategies that can be used to define concrete negotiation strategies. Using the EMF meta model as foundation allows us to define a graphical notation of the model.

A graphical notation of a negotiation strategy is shown in Figure 3.6 (page 74). Another negotiation strategy is visualized in Figure 4.10 that is used here to elaborate on the graphical notation in more detail. In the trapezoid shape the name of the negotiation object is mentioned. Thus, for different items that can be object of a negotiation, different trade-off strategies can be defined. This node serves as an artificial root of the trade-off strategy. Attributes that are negotiable are shown in boxes, labeled with their name. In the lower level of the box the domain for this attribute is shown, in curly brackets for an enumeration and in square brackets for interval values. The outgoing edges from the trapezoid are labeled with the priority of the sub-tree to which it connects. Edges between attributes represent trade-off pairs, these edges are labeled with the optimal trade-off-pair. Using such a graphical notion makes it reasonable for non-software developers to specify trade-offs for a conversation policy. Additionally, we can transform the trade-off model automatically into data-objects representing the strategy using MDD techniques. A detailed introduction in MDD is given by Steinberg et al. [SBPM09] and Stahl and Völter [SV06]. A detailed presentation of the idea of acquiring and representing trade-off strategies and case studies can be found in the work by Kurtanovic [Kur08] and Schumann et al. [SKT09].

[11]`http://www.eclipse.org/modeling/emf/`: Accessed 06/30/2010

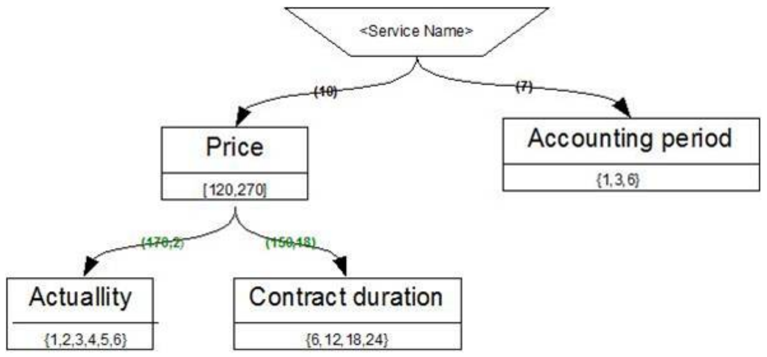


Figure 4.10.: Graphical of trade-off strategy, [SKT09].

Consequently, we can formally define trade-off policies and are able to express them in an intuitive way to humans. Thereby, their definition is based on a sound formal background.

Specification of conversation behavior

In the next process step conversation behaviors have to be specified. A conversation behavior is the behavior that defines how the agent behaves in reaction to a message that is part of a conversation. As pointed out before, the correct sequences of messages in a conversation is modeled by an automaton for each role. A conversation behavior defines the behavior in a state of such a conversation automata. For each state, a behavior has to be defined. Thereby, a conversation behavior can be used in different states. For example, the behavior in a final state of a conversation automaton implementing the roll-back if a negotiation is canceled. Typically, there exists a number of states in an automaton that represent an unsuccessful negotiation. And in each state the behavior can be the same. Each conversation state has incoming and outgoing edges. Exceptions are the initial state, that does not have any incoming edges and the final states that do not have any outgoing edges. Edges are labeled with an performative of an incoming/outgoing message or an ϵ-move. If a state is entered with an incoming message, like the state labeled with 2 in the contract net initiator conversation automaton shown in Figure 4.7 (on page 146), the conversation behavior has to handle the incoming message. Otherwise the state can be activated without an incoming message by an ϵ-move. If the outgoing edge of a state i is labeled with a message a, the conversation behavior has to determine the content of this message. If the current state of the conversation automaton has more than one outgoing edge, the conversation behavior has to determine which edge/action has to be taken. While computing the content of an outgoing message the conversation behavior has to respect the guidelines specified in the conversation policy. If the outgoing edge is an ϵ-move, the next conversation state is activated after the conversation behavior of the current state has been performed.

Conversation behaviors can be realized by an inference mechanism that uses the specification of the interaction protocol and the conversation policy. Goal-oriented or utility-based agents are capable of reasoning of the correct behavior. It is also possible that the conversation behaviors are designed and implemented by human developers. If the behaviors are manually developed, we recommend the usage of decision trees as implementations patterns.

A conversation behavior that is designed for a state with multiple outgoing edges can be designed in form of a decision tree. A decision tree takes a description of a situation as an input, given as a set of attributes, and computes a decision, a value that has been computed based on the input values (see Russell and Norvig [RN03, p. 653]). According to Witten and Eibe [WF05, pp. 62–65] a decision tree is defined as a tree with the following specific characteristics :

- inner nodes are labeled with attributes or conditions with the input attributes as terms.
- the outgoing edges of a node a are labeled with different possible outcomes of the conditions stated in a. In each state the condition is evaluated and the outcome should match to the value labeled at one outgoing edge. This transition is used to move to the next state, evaluating the decision tree.
- the leafs are marked with one element of the set of possible outputs.

Thereby, the decision tree is defined here in a way that the leafs of the tree specify the performative of the answering message, i.e., the outgoing edge of the conversation state. We illustrate this pattern by giving an example of a decision tree for the conversation behavior that can be applied in the state labeled with 2 in the contract net initiator conversation automaton shown in Figure 4.7 (on page 146). The conversation behavior has to determine if the agent should *accept* or *reject* the offer. A possible decision tree is shown in Figure 4.11.

Adaptation of existing planning systems

In this section we discuss different techniques that can be used to adapt existing planning system to enable the PAA to get access to the functionality of the local planning system. The reuse of existing planning systems is a critical issue for the coordination of planning systems in the industrial context. As pointed out in Section 2.2, those systems have been designed for a number of practical, i.e.,

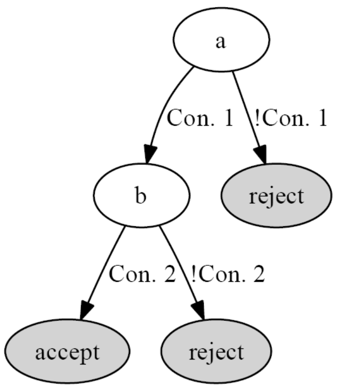


Figure 4.11.: Example of a decision tree, representing a conversation behavior. Con is the abbreviation for Condition

industrial, application fields. Those systems have been developed either as custom software or customized commercial software, see Stantchev [Sta07]. According to Hansen and Neumann [HN09, pp. 259,260] these software systems require significant investments and contains knowledge of the processes of the company have been embedded into those custom software systems. To gain more efficiency, it is desired to reuse those systems and not re-implement their entire functionality. Another aspect that emphasizes the adaptation of existing planning systems originate from the fact that a planning entity has to coordinate its activities in a number of networks. If each network would specify the planning methodology or the planning system that is interoperable within this particular network a planning entity would have to run a number of planning systems in parallel. These systems do not have to be interoperable with one another, and even if they are this would be inefficient, because of required redundancy of data and resources and the efforts for keeping all data current and synchronized. Although techniques have to be implemented to resolve conflicts resulting from different planning decisions. Consequently, the adaptation of existing planning systems seems to be the most realistic, and economic alternative. In the following we discuss how planning systems could be accessed by agent technology.

FIPA has proposed a draft for a standard for the agent software integration [FIP01a]. In this draft an explicit *wrapper agent* is defined that adapts existing software systems and offers their functionalities to other agents. They are capable to dynamically connect to existing resources, based on a description of the software system. Therefore, and for communication purposes FIPA has presented a special *Agent Resource Broker Ontology* and a *Wrapper Ontology* [FIP01a, pp. 9–16].

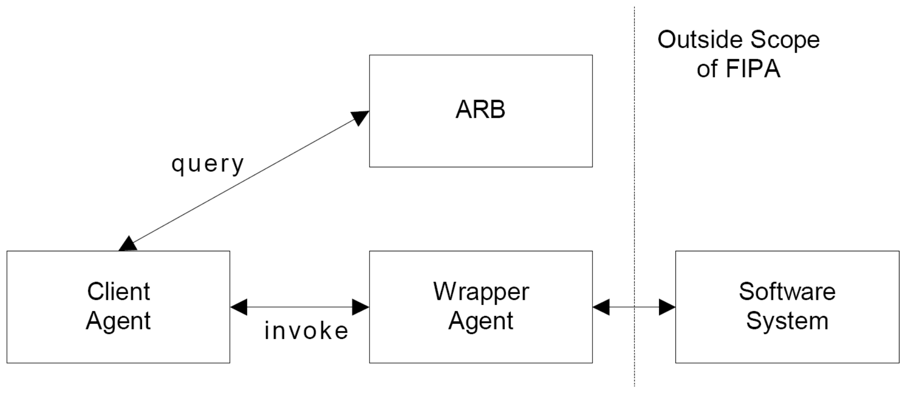


Figure 4.12.: General agent software integration scenario [FIP01a, p. 2]

Wrapper agents have to be able to communicate with this ontology to other agents. Client agents query for wrapper agents using an *agent resource broker* (ARB). This is sketched in Figure 4.12.

The wrapper agent maps a number of integration techniques like CORBA, DCom or Web Service to the ACL used for the communication between agents [FIP01a, p. 3]. The internal structure of the wrapper agent and used techniques for adapting existing software are not in the scope of the FIPA standard.

For summary, the FIPA draft specifies how agents access functionality of existing software by using a wrapper agent, but does not indicate how the wrapping of the software is done precisely.

Typically, it is not intended to allow all PAAs to access all planning systems. The access should be strictly limited to software controlled by the planning entity, as the local plans, planning knowledge and the planning process itself are typically private information of a company. Thus, the FIPA wrapper agent cannot provide its full potential, but it is still useful offering a unified interface to the PAA or in situations where no information hiding aspects are needed.

As the wrapping of legacy systems by the wrapper agent is not addressed by the FIPA draft, we are going to detail out *conventional* techniques for wrapping legacy systems. Mecella and Pernic [MP01] distinct between two different types of wrappers:

- Access wrappers: which provide a new interface that corresponds exactly to the one of the existing system, and

- Integration wrappers: which provide new interfaces that do not necessarily, are a direct mapping to existing software systems. An integration wrapper uses different access wrappers and additional logic to provide those new interfaces. Its goal is to present an integrated view of the underlying services.

As pointed out before, the interface to the framework has to be unified for different planning systems. Thus, the development of an integration wrapper is advocated, that also functions as the wrapping agent.

For realizing the access to the current planning systems we advocate to use web services. Web services have been identified a favorable technology for encapsulate existing functionality of software systems, see Sneed et al. [SWH10, p. 263]. The idea of using web services to implement an enterprise service bus to allow a broader application integration is widely accepted, see Papazoglou et al. [PTDL07]. A detailed introduction for web services and their implementation is given, e.g., by Zimmermann et al. [ZTP05, p. 2]. The encapsulation of functionality using web services is described in more detail by Sneed et al. [SWH10, pp. 272–279]. Web services can be integrated into agents, widely used agent-based frameworks like JADE offer add-ons like the Web Service Integration Gateway add-on[12], which enables calling web services easily within a JADE agent.

To do so, web services have to be developed that can be made accessible via the internet, or more realistically in the case of a planning entity to the local area network. Nevertheless, a web service only offers the interface to external applications. The logic of the wrapper has to be implemented, independent of the integration technology. For the implementation of wrappers different design patterns have been proposed. We present some basic design pattern here, all introduced by Gamma et al. [GHJV94]. Even those patterns have been discussed by the authors for object-oriented programming, they can be used similarly for the wrapping of existing systems. In the following description of the patterns we therefore refer to systems instead of objects, as this is more adequate for the scope of this study.

The first pattern discussed here is the adapter pattern, which is also known as wrapper pattern. The goal of an adapter is to change the interface of an existing system into another interface that is expected by its clients. If required the adaptor adds additional features. So it is more than a pure interface conversion but contains additional logic that is required to provide all functionality the client expects. For a detailed discussion with an exemplary implementation see Gamma et al. [GHJV94, pp. 137–150].

Another pattern that is referred to as wrapper is the decorator pattern, presented by Gamma et al. [GHJV94, pp. 175–184]. In contrast to an adapter pattern the decorator does not change the interface of a system, but adds addi-

[12]The add-on is available for the JADE platform at the web site `http://jade.tilab.com/community-addons.php`, Accessed: 03/15/2010.

tional functionality, that is needed.

The design of a wrapping web service can either be done top-down defining the interface, e.g., via the WSDL description, and then implementing the logic of the wrapper afterwards, or bottom-up designing the logic first and generating the web service description and the web service specific implementation.

Other middleware, as a technology for enterprise application integration, is not discussed here. A detailed introduction is provided by Ruh et al. [RMB01].

These design patterns and EAI technologies can be used to implement the adaptor for the wrapping agent, that offers the services of existing software systems to other agents.

4.3.3. CoPS framework

According to Sommerville [Som06, p. 427] a framework "is a sub-system design made up of a collection of abstract and concrete classes and interfaces between them." Its main purpose is to facilitate reuse of components that offers functionalities that can be reused for the development of systems. According to Sommerville [Som06, p. 427] frameworks can be classified into

- system infrastructure framework,
- middleware integration framework, and
- enterprise application framework.

As the CoPS framework is designed for a specific application domain, namely the coordination of planning entities, it can be classified as an enterprise application framework, which is based on some middleware integration framework, e.g., JADE in particular.

A framework offers either a set of interfaces and components, like abstract classes, that have to be implemented or instantiated to get use of the framework, or it can offer a set of components that have to be configured and can be used in the application to be designed. According to Gurp and Bosch [GB01] the first type of framework is referred to as white box framework, the later one as black box framework. The idea of a white box framework is to allow to adapt or to extend the framework at specific extension points, where the framework remains abstract, as the concrete implementation depends on the application at hand. Abstract classes are typically used to implement these extension points. According to Pree [Pre95, p. 253] these parts of the framework are called *hot spots*. By using hot spots the framework designer can specify which parts of the framework have to be detailed out for usage. In contrast to hotspots are the fixed parts of the framework. These parts of the framework cannot be changed, and are typically even not visible for the user of the framework. These parts of the framework are called frozen spots [Pre95, p. 253]. The idea of defining hot spots has been presented as a pattern for designing frameworks that is applied here, as well. By defining concrete hot spots we can allow to customize the agents to the given coordination task and the interest of each planning entity. The implementation of these hot spots is supported by the CoPS process that has been discussed above. On the other hand the usage of frozen spots that cannot be changed by the users of the framework can be used to implement some stable and expectable behavior of the agents. This design approach allows users of the framework the needed freedom to define domain specific parameters and policies on the one hand, and on the other hand restricts the agent's capabilities to a common core. Thus, the CoPS framework is a domain specialized agent-based framework.

The CoPS framework should ease the implementation of coordination mechanisms for the coordination of planning systems. Therefore, the CoPS framework offers an abstract implementation of the coordination agent and the planning authority agent and their interactions. As a consequence of the development process of the CoPS framework as part of this thesis, the framework is young. Moreover, it is not mandatory to use the CoPS framework if the ECo or CoPS process is applied. Therefore, the framework is not the foundation of the CoPS process, even though the CoPS process concerns specific aspects that have to be dealt with if the CoPS framework is used. Nevertheless, similar or the same steps have to be taken into consideration if the agents are implemented without the CoPS framework. Therefore, we see the CoPS framework as a supplement of the ECo and CoPS processes. In the following the CoPS framework is described in detail.

Overview of the CoPS framework

As pointed out above, the CoPS framework is a white box framework. The concrete agents have to be derived from the abstract classes of the framework. The CoPS framework is composed out of two major sub packages, namely the package for the coordination agent (`CA` package) and the planning authority agent (`PAA` package). Some, more general, concerns are dealt with in the overall framework package and some additional helper functionality is combined in an `util` package. The overall package structure is shown in Figure 4.13. As pointed out before, the CoPS framework is based on broader frameworks, in particular the JADE framework, for developing MAS in the Java programming language.

The packages concerning the coordination agent and the planning authority agent are presented in more detail in the following.

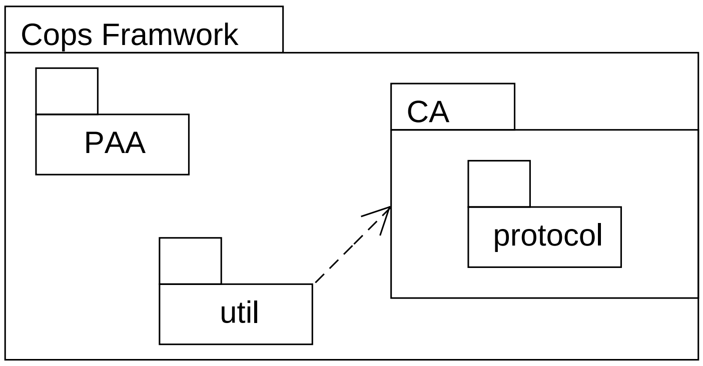


Figure 4.13.: Package diagram of the CoPS framework

The coordination agent

The specification of the coordination agent in the CoPS framework is discussed here. An extended class diagram of the current implementation of the `CA` package is shown in Figure 4.14. Note that this is an extended class diagram, as it contains a number of classes that are not part of the `CA` package. They are included to ease the understanding how the CoPS framework has to be used, and how it is based on the JADE framework.

The abstract class `Coordination Agent` inherits from the `Agent` class of the JADE framework (not shown in the diagram). As a key functionality of a coordination agent is to offer registration services for the PAA joining the network, we use the FIPA *Subscribe Interaction Protocol* [FIP02o], which has been implemented by the JADE framework. The interaction protocol is shown in Figure 4.15 to give the reader a quick overview of the specification of this interaction protocol.

Therefore, the classes `SubscriptionResonder`, `Subscription` and the interface `SubscriptionManager` from the `jade.proto` package have been added to the diagram, to visualize how the classes in the `CA` package are connected and how they are based on the JADE framework. The `CoordinationAgent` has an instance of a `CASubsciptionManager` as an attribute that implements the `SubscriptionManager` interface and therefore specifies what bookkeeping activities have to be performed if a PAA enters the network. This bookkeeping is done via the classes `MemberList` and `NetworkMember` in which information, like the offered capabilities of each network member are managed. Thus, while registering to a network the PAA sends a set of capabilities that he can perform as content of the subscription message for the network to the coordination agent.

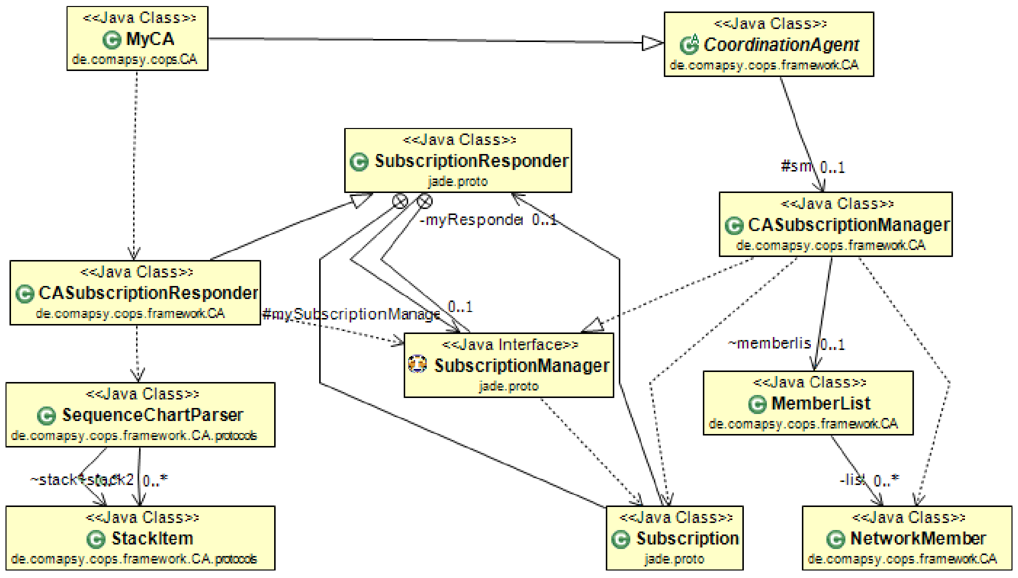


Figure 4.14.: Class diagram of the `framework.CA` package

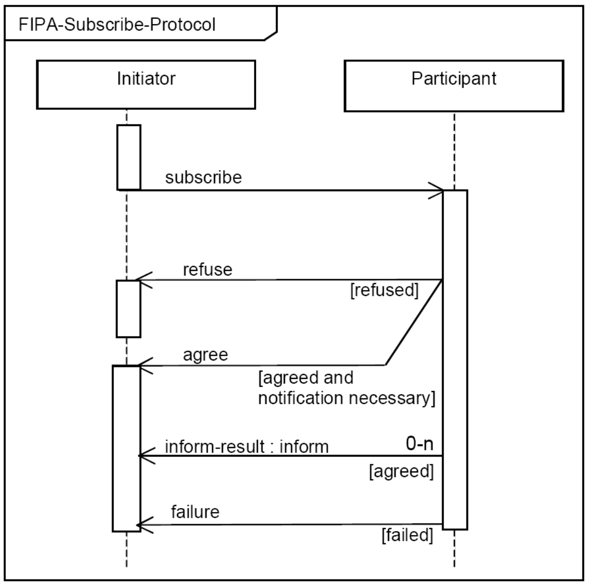


Figure 4.15.: FIPA Subscribe Interaction Protocol [FIP02o, p. 1]

The class MyCA is also not part of the framework. It demonstrates the usage of the CoPS framework for the design of a coordination agent. The MyCA class inherits from the abstract class CoordinationAgent. Moreover, it generates an instance of the CASubscriptionResponder class with specific parameters of the current network, like the network specific conversation protocol specifications, for instance. The corresponding class is defined in the framework and has to be initialized with custom parameters. In its constructor the conversation protocol specification is loaded and the conversation automata, discussed above, are generated. Therefore, the classes from the CA.protocol package, SequenceChartParser and StackItem are used. If an agent registers to an instance of the MyCA agent, the CASubscriptionResponder is activated and handles the subscription message. It generates an answer to the subscription with an AGREE performative. In the content of the answer, instances of the conversation automata are sent to the subscribing PAA.

The CA can be extended to enable anonymous communication between the network members, if we use alias names for the network members and all messages are send to the CA and then are forwarded to the agent that the message was intended to send to. The CA is designed to provide such features, but currently they are not implemented. This feature would be necessary for industrial use, but for the pure evaluation of the ECo-CoPS approach this feature does not add any value.

The planning authority agent

In this section, we present the design and the architecture of the PAA agent. In the design of the PAA we point out different extensions that can be applied to the PAA framework, and the extension points that have been build in. The design of the PAA is depicted in Figure 4.16. The agent design is based on a layered architecture, aspects like communication, internal reasoning, and interaction with external software systems are separated. This is motivated by the idea of the separation of concerns in software engineering. This separation also enables the control layer to operate on abstraction concerning the concrete implementation. The agent is designed in three layers. A communication layer, a control layer and an execution/wrapper layer for the integration of existing systems. This structuring of an agent can be found in the literature, e.g., Timm [Tim04, pp. 67–70], as well. Even though this is not a well-established agent architecture, like BDI, for instance, the layered approach should facilitate easier implementation for the separate sub-systems.

The task of the communication layer is to handle the ACL messages from and to other agents, to coordinate the activities. Therefore, it has to manage the network

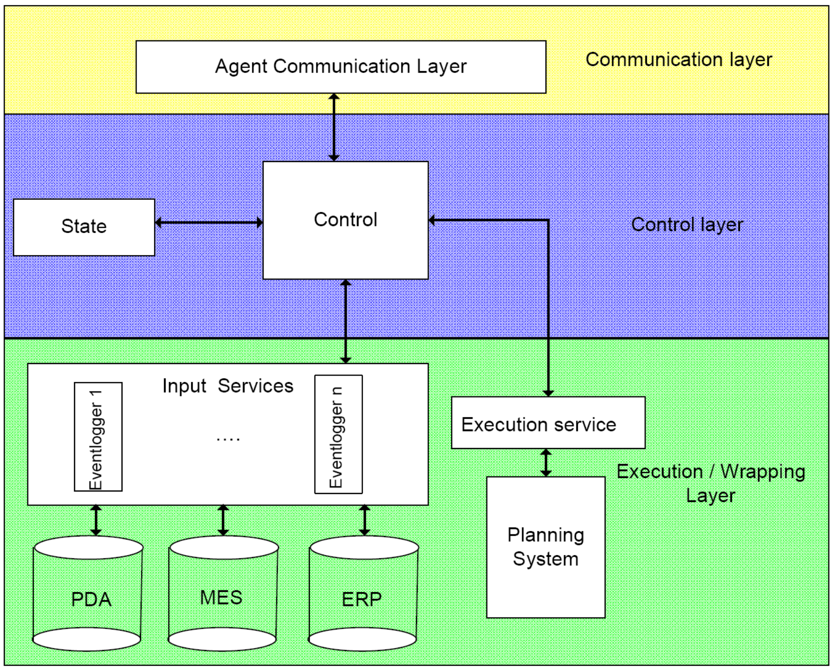


Figure 4.16.: Design of the PAA

specific protocols to use the correct interaction patterns in a given network. It is also responsible for adapting protocols with the customized conversation behaviors that have been specified and implemented by the planning entity.

The controller determines the activities of the agent. The reactive part of the agent's activities is triggered by communication, as request to evaluate the effects of a change or a request for scheduling some activity. A more pro-active part of the PAA is the search for local improvements. This search requires local knowledge, of course. As discussed above this knowledge should be embedded in an optional model of the wrapping service of the planning system. The controller can activate the search for local improvements. If such an improvement has been identified, the controller can try to initiate a negotiation with other agents, using the communication layer. The controller is the sub-system that updates the local state, which represents the environment and the current local plan. Thus, decisions concerning the local plan are triggered by the control flow in the controller, which uses additional information provided by the wrapping layer.

In the *execution and wrapping layer* different existing software systems that exist have to be addressed. Two aspects have to be mentioned in this layer. First, this layer has to adapt the local planning system. Thus, an execution service has to be defined that enables usage of the functionality of the existing planning system. Moreover, additional features that should be provided are the evaluation of a current plan or the search for modifications in the input of the planning system. The search for modifications could, for instance, relax a deadline that would allow generating a better plan according to the local evaluation function. The wrapping layer has to deal with the information integration of the PAA agent into the information flow within existing systems, as well. For instance, events that change the environment have to be identified and the controlling sub system of the agents has to be triggered. Therefore, interface to information collecting systems should be provided. Examples for these information collecting systems are production data acquisition (PDA) systems, manufacturing execution systems and enterprise resource planning systems. According to Kurbel [Kur05, Chap. 7] these systems can be sources for planning relevant information. Problems and possible techniques to identify planning relevant events in those systems have been discussed, e.g., by Schumann and Sauer [SS09]. Moreover, the current plan of the planning entity has to be communicated to the other information systems within the organization. One technique for realizing this, is to use the local planning system that has an additional representation of the local plan. But this approach has some disadvantages and makes it harder to evaluate potential plans, and options for future decisions, as one has to ensure that the current plan is not invalidated by these hypothetical computations. Another way doing so is to allow the PAA to ensure that the current plan is either send to other systems or stored in a joint database, that can be accessed by other systems, like the MES as well.

An extended class diagram of the current implementation of the `PAA` package is shown in Figure 4.17. Note that this is an extended class diagram, as it contains classes that are not part of the `PAA` package to ease the understanding how the CoPS framework.

A key element is the abstract class `PlanningAuthorityAgent` which is the base class for all implementations of localized PAAs. The class `MyPAA` is such an implementation of the PAA that inherits from the `PlanningAuthorityAgent` class. The `MyPAA` is not part of the framework, but an agent built on top of this framework.

The `PlanningAuthorityAgent` uses an instance of the `ListenerBehavior` which is derived from the `CyclicBehaviour` behavior of the JADE framework. As the name indicates this behavior is responsible for receiving all messages that have been sent to the agent. It passes these messages to an instance of the `CommunicationManager` class. In this class the different ongoing conversations of the PAA are managed, using the conversation automata and their customized conversation behaviors.

During the `setup` of the `PlanningAuthorityAgent` it executes the `SubscripeToNetworkBehaviour` inherited from the JADE `SubscriptionInitiator` class

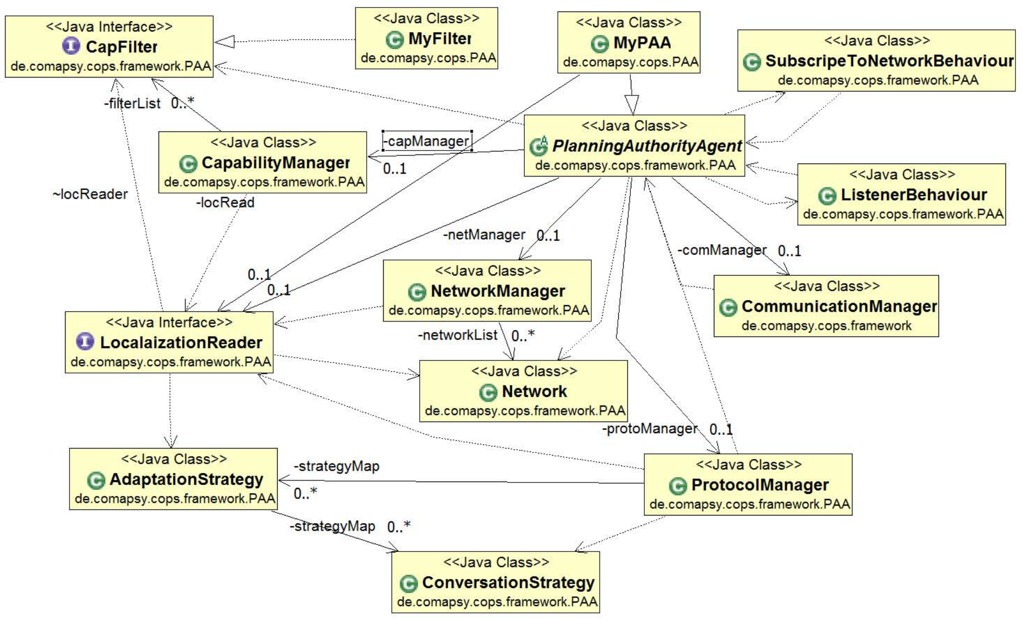


Figure 4.17.: Class diagram of the `framework.PAA` package

implementing the initiator role of the subscription protocol discussed above (see Figure 4.15). In this behavior the agent registers to an initial set of coordination agents. Of course, it can register later on to additional networks, or can de-register itself, if necessary. The agent manages network information in an instance of the `NetworkManager` class, which contains some instances of the `Network` class.

The capabilities of the planning entity, at least their representation in the coordination system, are managed by the `CapabilityManager`. As for each network the set of offered capabilities can be differed, for each network a filter has to be defined that maps the available capabilities of the planning entity to the subset that is offered to the specific network. This can be customized by implementing the interface `CapFilter`. This interface has been implemented by the `MyFilter` class, which is not part of the CoPS framework.

The customization of a PAA is a central issue for the usability of the framework. This customization information can be provided to the framework via the `LocalalizationReader` interface, which has to be implemented by each planning entity to provide the required local adaption. This interface provides information to the PAA like capabilities of the planning entity, networks it is participating in, and the adaptation strategies for the conversation automata provided by the coordination agents of these networks.

The information concerning the necessary adaption is given by a set of `AdaptationStrategy` objects. An `AdaptationStrategy` specifies for a number of networks the required adaptations. The adaption for a single network is described by an object of the `ConversationStrategy` class. Within such a strategy for each state of a conversation automaton, which is known at the design time, a conversation behavior can be specified. This conversation behavior has to be executed in that state of the automaton. Thereby, the designer can specify the local behavior within the conversation. The protocols that have been received after the registration are adapted according to the given strategy for the network and then are stored in a `ProtocolManager` object. If the agent wants to start a conversation with another agent, an appropriate protocol for the network can be retrieved from this object. More precisely, a new instance of an adapted conversation automaton for the specific role in the appropriate conversation protocol is provided by the `ProtocolManager` object.

The PAA framework is, up to now, not implemented completely. For example, the connectors to PDA systems are not realized. The core elements of the framework have been implemented.

5. Engineering coordination: The SPT case study

So far, we have presented the ECo process and the supporting CoPS process and the CoPS framework. In this and the following chapter the ECo-CoPS approach is validated. In this chapter we discuss the coordination problem of the first case study, the production of goods and their distribution. This scenario has been introduced in Section 2.1.1.

5.1. The SPT case study

Before we are going to execute the ECo process, we briefly describe the case study. For a more detailed description see Section 2.1.1. In the production process within a company goods have to be produced, packed on loading devices and then transported to the customers. Each of these steps defines a complex planning problem. The overall flow of material and workflow for each good is shown in Figure 5.1[1].

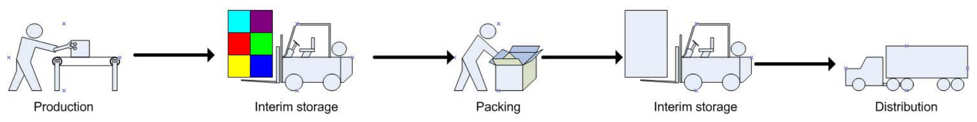


Figure 5.1.: Workflow of a the production and distribution example

We use existing planning systems that have been developed by students of the Goethe University Frankfurt am Main. The planning systems have been designed and implemented in the practical course "Praktikum Wirtschaftsinformatik und Simulation"[2] in the winter term 2009/2010, held by René Schumann and Prof. Dr. Ingo J. Timm. The task description that was provided to the students is presented in Appendix B.1. The planning systems have been implemented in Java and can

[1]This figure is the same as Figure 2.2. It is presented here again, to increase readability.
[2]Practical course "Business Informatics and Simulation"

be used as Web Services, based on the Tomcat[3] (version 6.0) and Axis2[4] (version 1.4.1) technology stack. As these systems have been developed by students, as part of their exercises in a one term course, the quality of the resulting plans should not be overrated.

We select and implement the needed coordination mechanisms using the ECo-CoPS approach. Therefore, we describe each step of the ECo process in a separate section.

5.2. Modeling the scenario

In this section we describe the first step of the ECo process, the modeling of the coordination problem. The context of this process step in the ECo process is shown in Figure 5.2.

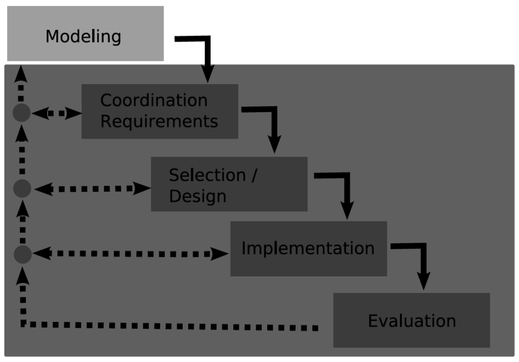


Figure 5.2.: Context of the modeling step in the ECo process

As already mentioned the case study has been presented in Section 2.1.1. Moreover, parts of the formalization have been presented in examples during the presentation of the ECo process (Section 4.2).

During the modeling of the coordination problem in this and the following chapter, a number of concepts, sets and functions are defined that are required to explain the problem space. To ease the reading we use some conventions for notation. For concepts, sets, and function we use different definition blocks to point out more clearly the subject of definition. Concepts are referred to in normal mathematical text, like c and a, for instance. Sets of concepts were encoded with calligraphic capitals like $\mathcal{C}$ and $\mathcal{A}$. Powersets are indicated by the notation $\wp(\mathcal{A})$. For the sets of numbers we use the notation $\mathbb{N}$ and $\mathbb{R}$ to refer to the sets of integer and real numbers. Numbers themselves were highlighted by encoding them sans serif, like x. Functions are identified by typing them in italic, like $f(x)$. All concepts, sets and functions introduced in this chapter are summarized in Appendix B.2.

As we here describe a dynamic system, the state of the system can changes over time. Time is represented as a discrete increasing integer value. Let t be a point in time (*pit*) with $\mathsf{t} \in \mathbb{N}$. So sets that can change over time can be described for every pit t. In this case study we assume that each time slot has a length of 15

[3]`http://tomcat.apache.org/`, Accessed: 03/19/2010
[4]`http://ws.apache.org/axis2/`, Accessed: 03/19/2010

minutes. Thus, a day can be described as 96 time intervals.

The task is to fulfill a number of orders, each requiring a certain good. Each good has to be produced, packaged and transported to the customer. An order is satisfied, if all goods have been delivered to the customer that belong to the order.

We now define a set of *exclusive* **resources** $\mathcal{R}$.

Set Definition 5.2.1

$\mathcal{R}$ is a formal language of labels for resources over an alphabet Σ.

A resource is reduced to a unique identifier. Resources are assumed to be non-shareable, and are required exclusively to execute an action. A single resource is specified with $r \in \mathcal{R}$. Resources are used by operations.

Concept Definition 5.2.1

An operation o is defined as $o = \langle r, \mathsf{d} \rangle$ with

- $r \in \mathcal{R}$, an identifier of the resource this operation requires
- $\mathsf{d} \in \mathbb{N}$, the duration the operation needs to be processed.

Based on the definition of an operation, we define the set of all available operations.

Set Definition 5.2.2

Let $\mathcal{O}$ be the set of all operations.

Based on this definition we can define a product p.

Concept Definition 5.2.2

A product p is defined as $p = \langle \mathsf{we}, \mathsf{h}, \mathsf{w}, \mathsf{d}, o_1, \ldots, o_n, <_o \rangle$ with

- $\mathsf{we} \in \mathbb{N}$, the weight of one unit of this product
- $\mathsf{h} \in \mathbb{N}, \mathsf{w} \in \mathbb{N}, \mathsf{d} \in \mathbb{N}$ the geometrical information of this product, as a box with height, width and depth.
- $o_1, \ldots, o_n$, with $o_i \in \mathcal{O}$ a finite sequence of operations that have to be performed to produce the product.
- $<_o$ a precedence relation defined for the operations.

Set Definition 5.2.3

The **set of all products** is referred to as $\mathcal{P}$.

Attributes of a product can be retrieved by projective functions, described in the following.

Function Defintion 5.2.1

For the product $p = \langle \mathsf{we}, \mathsf{h}, \mathsf{w}, \mathsf{d}, o_1, \ldots, o_n, <_o \rangle$ we define the following functions.

The function $weight : \mathcal{P} \to \mathbb{N}$ it is defined as $weight(p) = \mathsf{we}$.

The function $ops : \mathcal{P} \to \wp(\mathcal{O})$ it is defined as $ops(p) = \{o_1, \ldots, o_n\}$.

Another fundamental concept is the customer. A customer places orders and goods have to be delivered to the customers. Thus, her location is the relevant part for the modeling.

Concept Definition 5.2.3

A **customer** c is defined as $c = \langle \mathsf{x}, \mathsf{y} \rangle$ with $\mathsf{x}, \mathsf{y} \in \mathbb{N}$. The location of the customer is encoded by her x and y coordinates.

Set Definition 5.2.4

The **set of customers** is defined as $\mathcal{C}$.

Based on these concepts we can define an order.

Concept Definition 5.2.4

An **order** d is defined as $d = \langle p, \mathsf{q}, c, \mathsf{d}, \mathsf{e} \rangle$ The elements of this tuple are defined as follows:

- $p \in \mathcal{P}$, the product that has to be produced,
- $\mathsf{q} \in \mathbb{N}$, the quantity of the product that has to be produced,
- $c \in \mathcal{C}$, a customer,
- $\mathsf{d} \in \mathbb{N}$, due date, at that point in time the order should be finished, and
- $\mathsf{e} \in \mathbb{R}$, penalty that has to be paid to the customer per time unit of late delivery.

Set Definition 5.2.5

Let $\mathcal{D}$ be the **set of all orders**.

Attributes of an order can be accessed by projection functions.

Function Defintion 5.2.2

For the order $d = \langle p, \mathsf{q}, c, \mathsf{d}, \mathsf{e} \rangle$ we define the following function.

The function $quan : \mathcal{D} \to \mathbb{N}$ is defined as $quan(d) = \mathsf{q}$

The function $customer : \mathcal{D} \to \mathcal{C}$ is defined as $customer(d) = c$.

Scheduling
Packaging
Transport

Figure 5.3.: Dependency graph for the SPT case study

The function $due : \mathcal{D} \rightarrow \mathbb{N}$ is defined as $due(d) = \mathsf{d}$.

The function $pen : \mathcal{D} \rightarrow \mathbb{N}$ is defined as $pen(d) = \mathsf{e}$.

The overall problem can be described by a scheduling, a packaging and a transportation problem. Each of these problems is described in this case study in detail. For each of these problems a planning system, that can be accessed via a web service exists. In the following, we discuss the modeling of the different planning problems in more detail. After all planning problems have been discussed, some overall modeling issues are discussed.

The dependency graph, which is a linear one, is presented in Figure 5.3[5] .

Production scheduling

The production scenario has also been detailed in Section 2.1.1. There exist different operations, and for each operation two alternative resources are available. Transportation during the production process between these resources is neglected. This assumption has been introduced by Cavalieri et al. [CBM+99] for simplicity reasons, and is assumed here, as well. The shop floor layout has been presented in Figure 2.1 (page 14). The data for each production has been provided in Table 2.1 (page 14). The schedule has to be computed on the current set of all orders $\mathcal{D}$.

For the fulfilment of an order, it is necessary to produce the correct quantity of the specified product. These instances of products that have to be manufactured for a specific order are referred to as goods.

Concept Definition 5.2.5

A **good** g is defined as $g = \langle p, d \rangle$ with $p \in \mathcal{P}$ and $d \in \mathcal{D}$.

The production volume is determined by the number of goods that have to be produced.

Set Definition 5.2.6

The **set of goods** is referred to as $\mathcal{G}$.

[5]This graph has already been presented in Section 2.2.4.

Again we can define a projection function for goods in the following way.

Function Defintion 5.2.3

For the good $g = \langle p, d \rangle$ the following functions are defined.

The function *order* : $\mathcal{G} \rightarrow \mathcal{D}$ is defined as $order(g) = d$

The function *product* : $\mathcal{G} \rightarrow \mathcal{P}$ is defined as $product(g) = p$

The set of all goods of an order d can be referred to as $\mathcal{G}_d$.

Set Definition 5.2.7

The set of goods of an order d is defined as $\mathcal{G}_d \subseteq \mathcal{G}$, with

$$\forall g \in \mathcal{G} : g \in \mathcal{G}_d \Leftrightarrow order(g) = d$$

.

As it is required that for each order the correct number of goods is produced, it is necessary to identify the goods that belong to a specific order. This is done using the function *orderD*, which is defined as follows:

Function Defintion 5.2.4

The function *orderD* : $\wp(\mathcal{G}) \times \mathcal{D} \rightarrow \mathbb{N}$, is defined for the set $\mathcal{G}_d$ and the order d as follows $orderD(\mathcal{G}, d) = |\mathcal{G}_d|$

The set of goods $\mathcal{G}$ is *complete* if the following equation is satisfied.

$$\forall d \in \mathcal{D} : quan(d) = orderD(\mathcal{G}, d) \tag{5.1}$$

Based on the current set of goods a schedule has to be generated. For each good's operation specified in the product the corresponding actions have to be performed. An action is the planned execution of an operation.

Concept Definition 5.2.6

An **action** is defined as $a = \langle o, g, r, \mathsf{s}, \mathsf{e} \rangle$ with

- $o \in \mathcal{O}$ the operation that is performed,
- $g \in \mathcal{G}$ the good this actions belongs to,
- $r \in \mathcal{R}$ the resource the action is performed on,
- $\mathsf{s} \in \mathbb{N}, \mathsf{e} \in \mathbb{N}$ the start and end time of the interval the action is performed in.

The action is the smallest unit of a schedule. The set of all actions forms the schedule.

Set Definition 5.2.8

A **schedule** is the set of all actions it is referred to as $\mathcal{A}$.

Attributes of actions can be accessed to by projective functions, as well.

Function Defintion 5.2.5

For the action $a = \langle o, g, r, \mathsf{s}, \mathsf{e} \rangle$ the following functions are defined.

The function $good : \mathcal{A} \to \mathcal{G}$ is defined as $good(a) = g$.

The function $operation : \mathcal{A} \to \mathcal{O}$ is defined as $operation(a) = o$.

The function $start : \mathcal{A} \to \mathbb{N}$ is defined as $start(a) = \mathsf{s}$.

The function $end : \mathcal{A} \to \mathbb{N}$ is defined as $end(a) = \mathsf{e}$.

The function $resource : \mathcal{A} \to \mathcal{R}$ is defined as $resource(a) = r$.

As we pointed out, we are interested in the coordination of plans and schedules. Thereby, our goal is to ensure that these plans are feasible. Therefor it is necessary to define criteria indicating if a schedule is feasible.

First of all, it is necessary that the schedule is generated on the base of a complete set of goods, which has been encoded in Equation 5.1. To clarify additional aspects, we define a **partial schedule** for a good g.

Set Definition 5.2.9

A partial schedule for g is defined as $\mathcal{A}_g$, with:

$$\forall a \in \mathcal{A} : a \in \mathcal{A}_g : good(a) = g$$

.

In a partial schedule $\mathcal{A}_g$ all planned actions, and therefore, all operations that have to be scheduled to fulfill the order d are contained. For each good produced, all operations have to be performed in the sequence specified by the precedence relations. This is encoded in the following equations 5.2 and 5.3.

$$|ops(product(g))| = |\mathcal{A}_g| \wedge \forall a_1, a_2 \in \mathcal{A}_g, a_1 \neq a_2 : operation(a_1) \neq operation(a_2). \tag{5.2}$$

$$\forall a_1, a_2 \in \mathcal{A}_g, a_1 \neq a_2. \text{ Let w.l.o.g. } a_1 <_o a_2 \Rightarrow end_a(a_1) \leq start_a(a_2) \tag{5.3}$$

Another constraint, that has to be regarded is that resources are used exclusively, i.e., only one action per time is allowed to utilize the same resource. This is formalized as follows:

$$\begin{aligned} &\forall a_1, a_2 \in \mathcal{A} \neq a_1, a_2 : \\ &\nexists \mathsf{t} \in \mathbb{N} : resource(a_1) = resource(a_2) \wedge \\ &start(a_1) \leq t \leq end(a_1) \wedge start(a_1) \leq t \leq end(a_1) \end{aligned} \tag{5.4}$$

A partial schedule $\mathcal{A}_g$ for an order g is feasible, if the equations 5.2, 5.3 and 5.4 are satisfied. If all partial schedules are feasible and the resource usage constraint, described in Equation 5.4 is satisfied by the entire schedule, the overall schedule $\mathcal{A}$ is **feasible**.

To ease the following discussion we specify the concept of finished goods, which is in particular of interest for the following planning steps.

Concept Definition 5.2.7

A finished good f is defined as $f = \langle \mathsf{t}, g \rangle$, with

- $\mathsf{t} = \max_{\forall a \in \mathcal{A}g} end(a)$ the time the good is finished
- g the good

Set Definition 5.2.10

The set of **all finished goods** is referred to as $\mathcal{F}$.

The attributes of a finished product can be accessed by projective functions, as well.

Function Defintion 5.2.6

For the finished product $f = \langle \mathsf{t}, g \rangle$ the following function are defined.

The function $good : \mathcal{F} \rightarrow \mathcal{G}$ is defined as $good(f) = g$.

The function $ttime : \mathcal{F} \rightarrow \mathbb{N}$ is defined as $ttime(f) = \mathsf{t}$.

For easier handling of the following formula we define the following functions as abbreviations.

Function Defintion 5.2.7

The function $order : \mathcal{F} \rightarrow \mathcal{D}$, for $f = \langle \mathsf{t}, g \rangle$ is defined as $order(f) = order(good(f))$.

In the production data, shown in Table 2.1 (on page 14), a bottleneck occurs at resource three (see also Brennan and O [BO00]). Therefore, the group of students who have implemented the scheduling system favored a shifting bottleneck heuristic to solve the scheduling problem described in this section. A detailed description of this heuristic is given by Pinedo [Pin05, pp. 87–93]. The realized scheduler generates a valid schedule for a given set of orders and aims to maximize the usage of the given resources. The output of the scheduler is the set of finished goods $\mathcal{F}$. The production data described above is given to the scheduler. Furthermore, it is assumed that the production data is valid and deterministic and no resource breakdowns occur.

Packaging planning

For the packing of the goods, the set of finished products $\mathcal{F}$ is a necessary input. Moreover, information about the *bins* and the loading devices, is necessary. A loading device is characterized by its volume and costs induced per used instance.

Concept Definition 5.2.8

A **loading device** l is defined as $l = \langle \mathsf{m}, \mathsf{h}, \mathsf{w}, \mathsf{d}, \mathsf{c} \rangle$

- m maximal weight that can be loaded
- $\mathsf{h} \in \mathbb{N}, \mathsf{w} \in \mathbb{N}, \mathsf{d} \in \mathbb{N}$ geometrical information defining the volume that can be used per loading device
- $\mathsf{c} \in \mathbb{N}$ costs per used loading device

Set Definition 5.2.11

The set of all types of loading devices is referred to as $\mathcal{L}$.

We define projective function for loading devices for abbreviation issues.

Function Defintion 5.2.8

The function $cost : \mathcal{L} \rightarrow \mathbb{N}$. For $l = \langle \mathsf{m}, \mathsf{h}, \mathsf{w}, \mathsf{d}, \mathsf{c} \rangle$ it is defined as $cost(l) = \mathsf{c}$.

There exist some general restrictions concerning the number of loading devices that can be packed in parallel. The finished goods produced on one day have to be packed on loading devices at once. For simplification reasons we assume that the packing does not require any time. Thereby, a loaded device is defined as follows.

Concept Definition 5.2.9

A packed loading device pl is defined as $pl = \langle l, \mathcal{F}_{pl} \rangle$, with

- $l \in \mathcal{L}$ the type of used loading device and
- $\mathcal{F}_{pl} \subseteq \mathcal{F}$ the set of finished goods contained.

Set Definition 5.2.12

The set of loaded loading devices forms the **package plan** $\mathcal{PL}$.

Function Defintion 5.2.9

For a packed loading device $pl = \langle l, \mathcal{F}_{pl} \rangle$ the following functions are defined.

The function $items : \mathcal{PL} \rightarrow \wp(\mathcal{F})$ is defined as: $items(pl) = \mathcal{F}_{pl}$.

The function $mweight : \mathcal{PL} \rightarrow \mathbb{N}$ is defined as $mweight(pl) = mweight(l)$.

As it is assumed that the packing itself does not require any time, a packed loading device becomes available for transport if all items have been produced that should be placed in that particular loading device. This is, of course, a simplification that has been made to ease the tasks for the students. The time of availability of a packed loading device is computed by the function *available*.

Function Defintion 5.2.10

The function $available : \mathcal{PL} \rightarrow \mathbb{N}$. For $pl = \langle l, \mathcal{F}_{pl} \rangle$ it is defined as: $available(pl) = \max\limits_{f \in \mathcal{F}pl} ttime(f)$.

In a loading device a subset of the finished goods can be placed. Thereby, some restrictions have to be satisfied. The first restriction is that the total weight of the finished goods must not exceed the maximal allowed payload of the loading device.

$$\forall pl \in \mathcal{PL} : \sum_{f \in items(pl)} weight(product(f)) \leq mweight(pl) \tag{5.5}$$

Moreover, it is important that all items are placed correctly within the loading device. We enumerate these criteria and do not formalize them here, as it does not contribute to the understanding of the case study presented here.

Each item within a loading device is assigned to an anchor point that is used to identify the position of the item within the loading device. This anchor point is used to determine if the item has been placed within the device correctly. Finished goods within one device cannot be placed in a way that they would overlap. Moreover, each finished good must be placed completely within the inner borders of the loading device. Physical aspects have to be regarded, as well, like the fact that no finished good can float in the air. It must be placed either on the ground of the loading device or on top of another finished good. When stacking finished goods it is necessary to place the heavier items on the bottom and not on top of other finished goods.

If all these constraints are satisfied, a loading device is loaded correctly. Moreover, it is important that all finished goods are packed. The packaging plan is **complete** if the equations 5.6 and 5.7 are satisfied.

$$\forall f \in \mathcal{F} : \exists pl \in \mathcal{PL} : f \in items(pl) \tag{5.6}$$

$$\forall f \in \mathcal{F} : \nexists pl_1, pl_2 \in \mathcal{PL} : pl_1 \neq pl_2 : f \in items(pl_1) \wedge f \in items(pl_2) \tag{5.7}$$

If all loading devices are loaded correctly and the packing plan $\mathcal{PL}$ is complete the packing plan is *feasible*.

The planning problem that has to be described here is a 3-D bin packing problem with additional constraints regarding the weight of the objects. The students developed a heuristic that generates valid plans and tries minimize to number of loading devices required to pack all items. Thereby, the induced costs for loading devices are minimized, as the costs are linear to the number of devices used.

To give the transportation problem solver only the necessary information, we define the concept of *ready for shipment loading devices* rl. For each $pl = \langle l, \mathcal{F}_{pl} \rangle$ in $\mathcal{PL}$ a ready for shipment loading device is defined.

Concept Definition 5.2.10

A **ready for shipment loading device** is defined as $rl = \langle l, \mathsf{rt}, \mathcal{D}_{rl} \rangle$ with

- $l \in \mathcal{L}$ the loading device; equal to the corresponding pl.
- $\mathsf{rt} = \max\limits_{f \in \mathcal{F}pl} ttime(f)$ the earliest time the shipping can start
- $\mathcal{D}_{rl} = \bigcup\limits_{f \in \mathcal{F}pl} order(f)$.

Set Definition 5.2.13

The set f all ready for shipment loading devices is defined as $\mathcal{RL}$.

Function Defintion 5.2.11

For the ready for shipment loading devices $rl = \langle l, \mathsf{rt}, \mathcal{D}_{rl} \rangle$ the following functions are defined.

The function $rtime : \mathcal{RL} \to \mathbb{N}$ is defined as $rtime(rl) = \mathsf{rt}$.

The function $customer : \mathcal{RL} \to \wp(\mathcal{C})$ is defined as
$customer(rl) = \bigcup\limits_{d \in \mathcal{D}rl} customer(d)$.

The function $costLD :\to \mathbb{N}$ is defined as $costLD(rl) = cost(l)$.

The function $orders : \mathcal{RL} \to \wp(\mathcal{D})$ is defined as $orders(rl) = \mathcal{D}_{rl}$.

Transportation planning

The task of the transportation problem is to distribute the ready for shipment loading devices contained in $\mathcal{RL}$ to the customers they belong to. Therefore, a fixed number of trucks is available (maxNoofTrucks). A truck has a capacity for a number of loading devices. It starts from a depot and after it has served all customers it returns to the depot. Thus, the basic problem can be described as a vehicle routing problem. As pointed out above, the locations of customers are given as coordinates in a two dimensional Cartesian coordinate system, and the depot is located at the origin $(0, 0)$. Distances between two points can be computed using the Euclidean distance. A truck can be described as follows.

Concept Definition 5.2.11

A **truck** t is defined as $t = \langle \mathsf{ca}, \mathsf{s}, \mathsf{co} \rangle$, with

- $\mathsf{ca} \in \mathbb{N}$, the capacity of loading devices
- $\mathsf{s} \in \mathbb{N}$, the speed per time unit
- $\mathsf{co} \in \mathbb{N}$, the costs per distance unit

Trucks are assumed to have a constant traveling speed. Therefore, the needed time for traveling is linear to the distance between two points. If a truck arrives at a customer site, it has to stay there for a specified service time st.

Set Definition 5.2.14

The **set of all available trucks** is defined as $\mathcal{T}$.

Note that $|\mathcal{T}| = \mathsf{maxNoofTrucks}$ holds.

Function Defintion 5.2.12

For the truck $t = \langle \mathsf{ca}, \mathsf{s}, \mathsf{co} \rangle$ the following functions are defined.

The function $capa : \mathcal{T} \rightarrow \mathbb{N}$ is defined as $capa(t) = \mathsf{ca}$.

The function $speed : \mathcal{T} \rightarrow \mathbb{N}$ is defined as $speed(t) = \mathsf{s}$.

The function $costspU : \mathcal{T} \rightarrow \mathbb{N}$ is defined as $costspU(t) = \mathsf{co}$.

As part of the vehicle routing planning, loading devices have to be clustered together to be loaded on one truck and then the sequence of the customers within each tour has to be determined. The solution of the first sub-problem is called the truck load. A truck load is a set of ready for shipment loading devices.

Concept Definition 5.2.12

A **truck load** tl is defined as $tl = \langle \mathsf{t}, \mathcal{RT}_{tl} \rangle$, with

- $\mathsf{t} \in \mathbb{N}$ the earliest time this truck load can be shipped.
- $\mathcal{RT}_{tl} \subset \mathcal{RT}$ the loading devices loaded on the truck.

Note that the earliest time for the shipment of a truck load is determined by the time determined in the Equation 5.8.

$$\mathsf{t} = \max_{rl \in \mathcal{RL}_{tl}} rtime(rl) \tag{5.8}$$

Set Definition 5.2.15

The **set of all truckloads** is defined as $\mathcal{TL}$.

For a truck load the number of loading devices contained in this truckload can be retrieved by the function *noelements*.

Function Defintion 5.2.13

For the truckload $tl = \langle \mathsf{t}, \mathcal{RT}_{tl} \rangle$ the following functions are defined.

The function *noelements* : $\mathcal{TL} \rightarrow \mathbb{N}$ is defined as $noelements(tl) = |\mathcal{RT}_{tl}|$.

The function *elements* : $\mathcal{TL} \rightarrow \wp(\mathcal{RT})$ is defined as $elements(tl) = \mathcal{RT}_{tl}$.

The function *rtime* : $\mathcal{TL} \rightarrow \mathbb{N}$ is defined as $rtime(tl) = \mathsf{t}$.
The function *customers* : $\mathcal{TL} \rightarrow \wp(\mathcal{C})$ is defined as $customers(tl) = \bigcup_{rt \in \mathcal{RT}tl} customer(rt)$.

The function *orders* : $\mathcal{TL} \rightarrow \wp(\mathcal{D})$ is defined as $orders(tl) = \bigcup_{rt \in \mathcal{RT}tl} orders(rt)$.

For each daily production, the trucks have to deliver the goods to the customers. Therefore, truckloads are assigned to trucks. This assignment is defined as follows.

Concept Definition 5.2.13

A **loading assignment** is defined as $la = \langle \mathsf{s}, \mathsf{e}, tl, t, tour \rangle$, with

- $\mathsf{s} \in \mathbb{N}, \mathsf{e} \in \mathbb{N}$ the start and end time of the tour that distributes the truck load
- $tl \in \mathcal{TL}$ the truck loading that has to be distributed
- $t \in \mathcal{T}$ the truck that is assigned
- *tour* a sequence of locations to visit. The first and the last point in that tour have to be the depot.

Set Definition 5.2.16

The set of all loading assignments is called $\mathcal{LA}$.

Function Defintion 5.2.14

For the loading assignment $la = \langle \mathsf{s}, \mathsf{e}, tl, t, tour \rangle$ the following functions are defined.

The function *load* : $\mathcal{LA} \rightarrow \mathcal{TL}$ is defined as $load(la) = tl$.

The function *truck* : $\mathcal{LA} \rightarrow \mathcal{T}$ is defined as $truck(la) = t$.

The function *start* : $\mathcal{LA} \rightarrow \mathbb{N}$ is defined as $start(la) = \mathsf{s}$.

The function *end* : $\mathcal{LA} \rightarrow \mathbb{N}$ is defined as $end(la) = \mathsf{e}$.

The length of the tour with n elements in the sequence can be computed by the following function.

Function Defintion 5.2.15

The function $length : \mathcal{LA} \rightarrow \mathbb{N}$. For $la = \langle \mathsf{s}, \mathsf{e}, tl, t, tour\rangle$ it is defined as

$$length(la) = \sum_{i=1}^{i=n-1} : dist(c_i, c_{i+1})$$

, with c_i and c_{i+1} two subsequent elements of the $tour$.

The function $dist(c_1, c_2)$ computes the Euclidian distance between the points specified by the two customers. Note that for each loading assignment the Equation 5.9 must hold.

$$end(la) > start(la) + length(la) \tag{5.9}$$

It is a strict greater than and not a greater or equal relation here because at each customer side a stop for at least the service time (st) has to be made. Consequently, the duration for a tour can be computed by the function $duration$.

Function Defintion 5.2.16

The function $duration : \mathcal{LA} \rightarrow \mathbb{N}$ is defined for $la = \langle \mathsf{s}, \mathsf{e}, tl, t, tour\rangle$ as $duration(la) = noelements(tl) * \mathsf{st} + length(tour) * speed(t)$.

Another, more complicated aspect, is to compute the time of service of a customer.

Function Defintion 5.2.17

The function $serviceTime : \mathcal{C} \times \mathcal{LA} \rightarrow \mathbb{N}$ is defined for $la = \langle \mathsf{s}, \mathsf{e}, tl, t, tour\rangle$ as $serviceTime(c, la) = \mathsf{s} + duration'(la)$

The duration is computed for the sub-tour starting from the depot to the customer c. If customer c is not part of the tour the function returns 0. All trucks have a time window per day in which they are available. Thus, the start and end time for each load assignment have to respect those boundaries $\mathsf{earliest\_start}$ and $\mathsf{latest\_end}$, as well.

An loading assignment is *feasible* if the following equations are satisfied.

$$\forall la \in \mathcal{LA} : noelements(load(la)) \leq capa(truck(la)) \tag{5.10}$$

$$\forall la \in \mathcal{LA} : start(la) \geq rtime(load(la)) \tag{5.11}$$

$$\forall la \in \mathcal{LA} : \forall c \in customers(load(la)) : c \in tour(la) \tag{5.12}$$

$$\forall la \in \mathcal{LA} : end(la) - start(la) \geq duration(la) \tag{5.13}$$

$$\forall la \in \mathcal{LA} start(la) \geq \mathsf{earliest\_start} \tag{5.14}$$

$$\forall la \in \mathcal{LA} start(la) + duration(la) \leq \mathsf{latest\_end} \tag{5.15}$$

In Equation 5.10 it is stated that trucks are not allowed to be overloaded. In Equation 5.11 it is declared that the start time of a tour has to be greater or equal to the time the truck load is ready for transport. For each loading assignment all customers contained have to be served by the truck (Equation 5.12). In Equation 5.13 it is declared that within the time window of a tour all customers have to be served. Finally, in the Equations 5.14 and 5.15 it is pointed out that the time the truck is available are respected.

A truck can be assigned to a number of loading assignments. The assignments for one truck are the plan of the truck.

Set Definition 5.2.17

The **plan of a truck** t is defined as $\mathcal{LA}_t \subset \mathcal{LA} : \forall la \in \mathcal{LA} : la \in \mathcal{LA}_t \Leftrightarrow truck(la) = t$

A plan of a truck is feasible iff $\forall la \in \mathcal{LA}_t$ are feasible and additional Equation 5.16 is satisfied, that ensures that the assignments are not overlapping.

$$\forall la_1, la_2 \in \mathcal{LA}_t : \nexists t : start(la_1) \leq t \leq end(la_1) \wedge start(la_2) \leq t \leq end(la_2) \quad (5.16)$$

The set of all loading assignments $\mathcal{LA}$ is feasible if all truck plans are feasible and all ready for shipment loading devices are assigned to exactly one ($\exists!$) loading assignment. This is encoded in Equation 5.17.

$$\forall rl \in \mathcal{RL} : \exists!\ la \in \mathcal{LA} : rl \in elememts(load(la)) \quad (5.17)$$

Another relevant partitioning of $\mathcal{LA}$ is the partitioning according to orders that are served.

Set Definition 5.2.18

The set $\mathcal{LA}_d \subseteq \mathcal{LA}$ is defined as $\forall la \in \mathcal{LA} : la \in \mathcal{LA}_d \Leftrightarrow d \in orders(load(la))$

The students developed a heuristic for generating a feasible tour plan thereby trying to minimize the costs that occur during the delivery. These can be achieved by

- avoidance of penalty costs, due to too late delivery
- minimization of traveling distance, to reduce traveling costs.

5.3. Coordination requirements

The second step of the ECo process is the definition of requirements. The context of this process step in the ECo process is shown in Figure 5.4.

The coordination requirements in this case study are rather simple. The main objective is to ensure that the overall plan and all partial plans are feasible. The criteria for feasibility for the partial plans for scheduling ($\mathcal{A}$), for packaging ($\mathcal{PL}$) and for transportation ($\mathcal{LA}$) have been discussed already. But it is not sufficient that the partial plans are feasible. The resulting overall plan does not have to be consistent, even if all partial plans are. Aspects like the fact that all goods have to be produced before they can be packed cannot be covered by local feasibility criteria. Therefore, we have to detail here the specification towards additional feasibility characteristics. To ensure feasibility of the overall plan, two interfaces have to be considered. The interfaces from scheduling to packaging planning and from packaging planning to transportation planning. In the following we discuss these interfaces in detail.

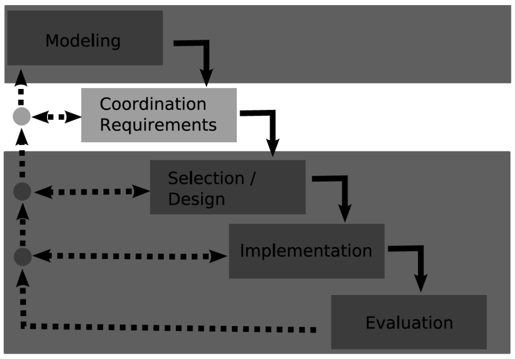


Figure 5.4.: Context of the requirement definition step in the ECo process

At first, we discuss the interface between the scheduling and the packaging planning. The input for the packaging planning is $\mathcal{F}$. For achieving an overall feasible plan, $\mathcal{F}$ has to be *complete* and *consistent*. By completeness we mean that all goods that are necessary to fulfill the orders are contained. Assume $\mathcal{G}$ is complete, according to the conditions stated in Equation 5.1. If $\mathcal{G}$ is complete then $\mathcal{F}$ is complete if it satisfies Equation 5.18.

$$\forall f \in \mathcal{F} : good(f) \in \mathcal{G} \tag{5.18}$$

Moreover, $\mathcal{F}$ has to be consistent. $\mathcal{F}$ is consistent if all goods have been completely produced, i.e., all their actions have been performed. Note that in the definition of f (Concept Definition 5.2.7) it has been defined in a way that its earliest time for disposal is defined as the maximum of the end times of the activities related to the good. Thereby, $\mathcal{F}$ is consistent by definition of f.

Consequently, if the planning system has to ensure that $\mathcal{F}$ is consistent and complete. The packaging planning can generate a plan that can be executed sequentially with the production scheduling plan. The planning system that has been developed by the students creates complete and consistent plans. Its results are in similar form as the definition of finished goods indicate. All data exchange formats can be found in the Appendix B.1, as well.

The second interface that has to be investigated in more detail is the interface between the packaging planning and the tour planning. The input for the transportation planning is the set of ready for shipment loading devices $\mathcal{RL}$. $\mathcal{RL}$ has to be complete and consistent, to ensure feasibility of the overall plan. Completeness characterize that all finished goods have been packed on a ready for shipment loading device. As pointed out before, the transformation into rt is done for interfacing the transportation planning. The internal packaging plan $\mathcal{PL}$ has been defined concerning completeness in an appropriate manner (see equations 5.6 and 5.7). For each element of $\mathcal{PL}$ a ready for shipment loading device is defined. Therefore, if it can be stated that if $\mathcal{PL}$ is complete and the projection is correct then $\mathcal{RL}$ is complete, too.

The second aspect that has to be addressed is consistency. Consistency characterize that no loading device can be transported before it is not packed. As packaging is assumed to take no time, this is equivalent to the fact that each finished good has to be produced before it can be shipped. As pointed out before, the time when rl can be shipped is defined as

$$\mathsf{rt} = \max_{f \in \mathcal{F}pl} ttime(f).$$

Thus, consistency is guaranteed by the definition of rt (see Concept Definition 5.2.10). If information concerning $\mathcal{F}$ and $\mathcal{RL}$ is computed correctly, and this information is processed correctly in the *consuming* planning system, the overall plan is feasible.

So far, we have defined the criteria for the overall plan to be feasible. In the reminder of this section we define the evaluation function for the quantitative evaluation and other non-functional requirements.

First, we tackle the aspect of further requirements. In this simple scenario were no information hiding or other special issues occur, only one aspect remains to be mentioned. As there exist already planning systems that are able to generate feasible local plans, these planning systems should stay in use. Ideally they should be used in the form of black boxes by the coordinating agents. Moreover, as the ECo-CoPS approach addresses the coordination of planning entities, it is the most reasonable approach to keep the different autonomously acting planning systems in place and do not replace them. This replacement is not trivial at all, because Defining the problem in form of an optimization problem is not trivial, and a sufficient model currently does not exists. Note that we not present one in this study, as this is not in our scope.

The objective function for the entire production scenario is to minimize the overall costs induced by producing and distributing the goods. No cost information for the production system has been given. Thus, in this model, costs are caused by packaging and distribution. A critical issue in determining the overall costs is the computation of the penalties that have to be paid for late delivery. To define the penalty, we first have to define the fulfillment time of an order. As already mentioned, an order is fulfilled if all goods have been delivered.

Function Defintion 5.3.1

The function $ft : \mathcal{D} \times \wp(\mathcal{LA}) \rightarrow \mathbb{N}$ is defined for c, d and $\mathcal{LA}_d$ as $ft(d, \mathcal{LA}_d) = \max_{la \in \mathcal{LA}_d} serviceTime(c, la)$.

Based on this definition we can define the penalty.

Function Defintion 5.3.2

The function $penalty : \mathcal{D} \times \wp(\mathcal{LA}) \rightarrow \mathbb{N}$ is defined for d and $\mathcal{LA}_d$ as

$$penalty(d, \mathcal{LA}_d) = \begin{cases} 0 & ft(d, \mathcal{LA}_d) \leq due(d) \\ (ft(d, \mathcal{LA}_d) - due(d)) * pen(d) & \text{otherwise} \end{cases}$$

Thus, the objective function can be defined as shown in Equation 5.19. It consists of three parts: the costs for packaging, the costs for transportation, and the costs for penalties.

$$\begin{aligned} cost = & \sum_{rl \in \mathcal{RL}} costLD(rl) + \\ & \sum_{la \in \mathcal{LA}} costspU(truck(la)) * length(la) + \\ & \sum_{o \in \mathcal{D}} penalty(d, \mathcal{LA}_d) \end{aligned} \tag{5.19}$$

5.4. Selection of appropriate coordination mechanisms

In the third step of the ECo process applicable mechanisms are identified. The context of this process step in the ECo process is shown in Figure 5.5.

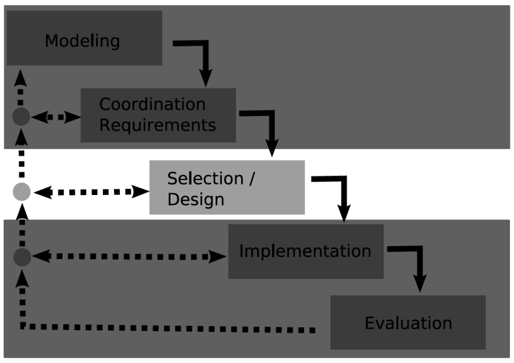


Figure 5.5.: Context of the selection step in the ECo process

In the following, we discuss the coordination methodologies identified as applicable for this scenario in Section 4.2.3 (on page 130). Then we present the suitable approaches that can be implemented and evaluated in this study.

- plan merging,
- decentralized planning for a centralized plan,

- result sharing, and
- negotiation.

Plan merging

As pointed out in Section 3.1.2, the main idea of plan merging is that a centralized entity collects the locally generated plans and ensures feasibility of the plan, if they are executed together. For implementing the centralized entity it is necessary that this entity is aware of dependencies between planning systems, and that it is capable to resolve existing conflicts without introducing new conflicts. Therefore, it is necessary to model detailed criteria for feasibility and planning knowledge into this merging entity. Consequently, this approach will result in a system similar to a centralized planner, which we argued, is not of interest in this study.

Decentralized planning for a centralized plan

The idea of different domain experts creating a shared plan, as presented in Section 3.1.2 and outlined by Durfee [Dur99, p. 141] seems to be adequate for the scenario at hand. Different planners are responsible for different parts of the plan. Durfee did not present any details how the overall plan should be kept consistent/feasible. In the examples presented by Durfee, each planning expert defines its part of the plan and then passes the results to the next planner. This corresponds to a coordination of the partial plans using a result sharing mechanism. The sequence of planners is thereby defined by the dependency graph that has been presented in Figure 5.3. Result sharing as a coordination mechanism is discussed below.

Result sharing

The exchange of information concerning existing plans will be part of any reasonable solution for this case study. As it has become clear during the modeling process, the output data of one planning system is a mandatory input for the following system in the dependency structure. The most simple technique for result sharing is, as pointed out previously, the sequential execution of each planning system and then passing the results to the next planning systems in the dependency graph. This strategy would result in first generating a plan for the production, then for the packaging and finally for the transportation sub-problem. As no cyclic dependency exists in this case study this seems to be a valid approach to achieve a feasible overall plan. In fact the students in the practical course, who had to deal with the problem described, have chosen this approach.

A criticism to this coordination mechanisms is that even if the resulting plans are feasible, their quality in terms of the objective function, the costs induced while fulfilling the orders do not necessarily have to be low. The largest portion of the costs are fixed during the transportation planning, even though they might not be caused by poor transportation planning. This is caused by the fact that during the scheduling and partly during the packaging planning no information about effects of local decision making on the global evaluation function is available. Even if this information would be available, the local planning systems have objectives that would not take advantage of it, because it does not lead to optimal local plans according to their own local objective function. So, pure result sharing in sequential planning can lead to global solutions of lower quality.

Negotiation

Another idea to tackle the problems is to start the planning first with the part that determines the most costs, i.e., the transportation planning. This corresponds to the idea of backward planning (discussed already in Section 3.1.1). This approach makes it more complex to ensure consistency/feasibility of the overall plan, as the sequence of executing planning systems is inverted according to the dependency graph, shown in Figure 5.3.

A solution to this problem can be a mechanism that facilitates the exchange of requirements towards the local plans, and plan suggestions that tries to satisfy the requirements and still ensuring feasible local plans. Such a coordination approach would result in a sequence of exchanges of requirements to, and suggestions of plans. This corresponds to a negotiation, trying to minimizing the total costs. By starting with the parts of the planning process where most of the costs are fixed requirements can be identified that lead to an overall solutions with lower costs. Previous planning stages have to identify what requirements are possible to fulfill and offer those to the subsequent planning entity.

In summary, we state that mechanisms using plan merging might not be possible. The overall problem of the SPT case study is a type of distributed planning with a centralized governed execution. Both result sharing and negotiations can lead to an overall feasible plan, but might with different outcomes, concerning the overall quality of the generated plans.

Qualitative evaluation

In this qualitative evaluation the compliance of the discussed coordination mechanisms to the coordination requirements is addressed in detail. Plan merging has been excluded already, because it would result in building a centralized planning system, and violate the requirement that existing systems should stay in use. Coordination by sequential planning with result sharing, and the approach of finding compromises about requirements from subsequent planning entities and feasible solutions of preceding entities are discussed in the following.

For the discussion of the compliance to the coordination requirement, one assumption has to be made. It is assumed that the planning systems in charge work correctly in the sense that with a given input they generate a feasible local plan that is represented correctly as the output of the planning system. The internal mechanisms and functionalities of the planning systems are not part of this study, they are pre-existing.

To ensure that the coordination requirements are satisfied the following aspects have to be discussed.

- Are the local plans feasible?
- Do the overall plan stay feasible?
- Stay existing planning systems in use?

As already mentioned, both approaches aim at using existing planning systems, and those systems generate only feasible local plans. Thus, two out of three requirements can be assumed to be satisfied by both approaches. Thus, only one aspect has to be discussed here in detail.

Before we discuss the approach of result sharing and sequential planning. The results of the planning stages, namely the set of finished goods $\mathcal{F}$ and the packaging planning $\mathcal{RL}$, have to be complete and consistent to ensure an overall feasibility. We assumed that both plans are the correctly computed by the planning systems. Those sets are passed directly to the packaging and transportation planning, and are used as inputs. The overall plan is feasible, if the planning systems work correctly.

Before we discuss the consistency of the negotiation approach, we briefly describe the procedure of this approach. Most of the costs are generated in the distribution and packaging planning. Moreover, planning relevant data for the scheduling system cannot be changed, as there exists no earliest start time for orders, for instance. So, what can be changed is the input data of the packaging and transportation planning. In particular, the time when products are released to be packed and transported can be shifted into the future without inside knowledge of the planning systems. In this coordination process the set of combinable orders has to be computed at first. Two orders are combinable if it is reasonable for the packaging planning to load their finished goods on the same loading device. Therefore, a packaging plan is computed that assumes that all products are available at the same time, to facilitate most efficient packaging planning. Orders are packed together on one loading device are combinable orders for each other. In the following we encode this set for order d as $\mathcal{D}^c_d$.

Next for each order the costs of the round trip tour are computed. This can be done with the existing planning system, by generating appropriate input data and interpreting resulting outputs. Thus, we have to made this information available. We do so in form of the following function.

Function Defintion 5.4.1

The function *rtdistance* is defined as *rtdistance* : $\mathcal{D} \rightarrow \mathbb{R}$. For d it is defined as $rtdistance(d) = \mathsf{x}$ whereby x is the distance computed by the transportation planning.

With this additional information, the economic efficient combinable orders ($\mathcal{D}_d^{ec}$) for an order d can be computed. An order d_1 is economic efficient combinable orders with order d iff both orders are combinable orders and the distance (and thus the transportation costs) of a joint tour is smaller or equal to the sum of both round trips tours. This is, of course, a heuristic, as a set of orders can become an efficient combination, even if they are not, in a pairwise comparison. Then the latest release time for each order is computed, that would allow to

- deliver time before the official due, and
- combine items of combinable orders within the packaging planning.

We compute the latest acceptable ready time for packaging for each order. Therefore, the finished products are grouped by days. The latest acceptable ready time is the latest time a product is finished before the due date of the order. An order is delayed to either its own latest acceptable ready time or the latest acceptable ready time of one of its economic efficient combinable orders with a ready time before its due. Based on this idea, the latest point an order can be delayed to can be computed by a fixed point iteration. We delay the finished products of an order that are ready for packaging and shipping before that latest acceptable ready time. Thereby, each product is deferred to the first time unit of the following day. This is done for simplification reasons, as the entire production of one day is loaded on loading devices (which is assumed to take no time) and then is distributed on the following day.

For this reason the official ready time is smaller than within the delayed data for each finished product. This guarantees that the interface between production and packaging planning systems is kept consistent, because neither the number of products nor their product-types are changed at all.

The final production schedule is generated by a run of the scheduling system. Thereafter, the input data for the packaging planning is changed as described above. Then packaging and transportation planning are done sequentially. Thus, within the planning sequence the only aspect that we have to discuss is the manipulation of the input data of the packaging planning, as this is the only aspect that is changed in comparison to the planning sequence discussed in the first approach.

The interface between the packaging and transportation planning is not changed for this planning approach, and therefore, the consistency and completeness aspects discussed above are valid her as well.

5.5. Implementation of coordination mechanisms

In this section we discuss the fourth step of the ECo process. In this step we discuss implementation issues concerning both coordination approaches. The context of this process step is given in Figure 5.6.

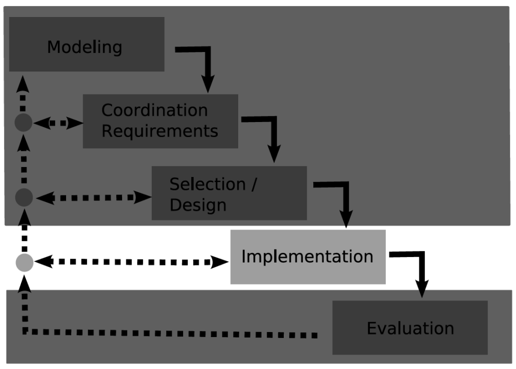


Figure 5.6.: Context of the implementation step in the ECo process

The result sharing approach is based on the idea of forwarding planning results. This approach has been implemented by the students of the practical course. The input data for the planning systems for packaging and transportation have data requirements, concerning the production schedule and the packed loading devices. This data is required in addition to information that is provided to all planning systems, like orders or products. The data exchange formats are presented in Appendix B.3. In our validation, we use the implementation that has been designed by the students as a reference for the second approach.

In the following the implementation of the negotiation-based approach is discussed. Therefore, we apply the CoPS process to describe the implementation. First the specification of needed conversation protocols has to be detailed. For specifying the conversation protocol the *FIPA Request Interaction Protocol Specification* [FIP02l] is used. The interaction protocol is shown in Figure 5.7.

As pointed out before, the agents try to achieve a common goal. They are cooperative and truth telling, as they are part of one legal entity and an overall objective function exists. The specification of the conversation protocol can be described as follows. The requesting agent sends its requirements to the preceding planning entity. If an agent receives a request, it has to accept the request and has to try to fulfill the requested specification within its local plan. Therefore, it has to find inputs, i.e., requirements, of its local plan, that satisfies the requested requirements. If it can achieve at least a partial fulfillment of the requirements it sends an inform message, containing the new relevant information for the subsequent planning entity. If it cannot achieve a fulfillment of the requirements it sends a failure message. The conversation policies of the local planning entities are

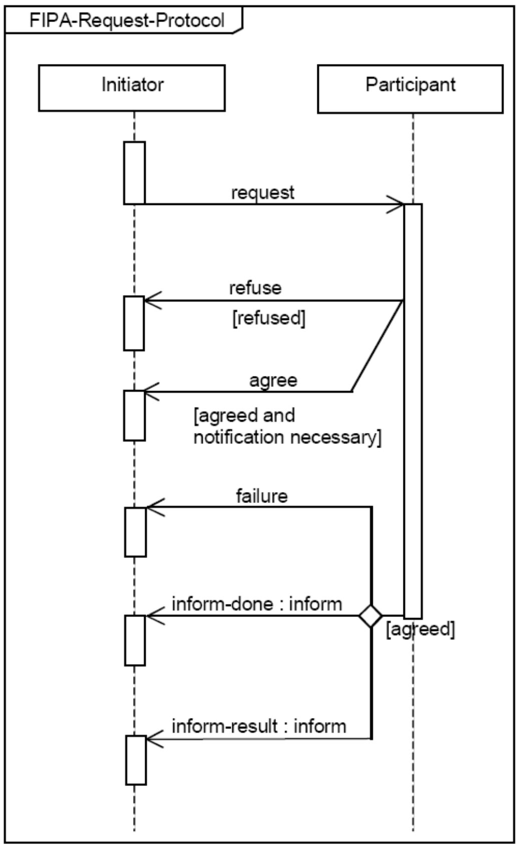


Figure 5.7.: FIPA Request interaction protocol [FIP02l, p. 1]

simple here. There are no tactical considerations, as all planning entities belong to the same legal entity. If a request comes to the entity, the entity tries to satisfy it.

In our prototypical implementation we accessed the local planning systems via prepared scripts that agents can execute, and web services executed by the agents themselves. For an implementation for industrial application, we advocate for the usage of dedicated wrapping agent for each planning system. A wrapper agent offers the planning service to the corresponding coordination agent, which is responsible for the coordination process. Therefore, the wrapping presented here corresponds to the FIPA wrapping standard [FIP01a], also discussed above in Section 4.3.2.

5.6. Evaluation of coordination mechanisms

In the following section we describe the final step of the ECo process, the evaluation. The context of this process step in the ECo process is given in Figure 5.8.

Within the evaluation the results of a comparison between a sequential planning and the improved planning procedure are presented. Therefore, we have developed a generator that automatically creates random problem instances. In the study performed here, most of the parameters describing a problem instance are kept constant, only the orders are subject of change.

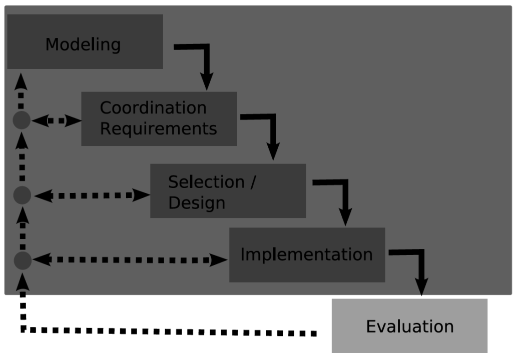


Figure 5.8.: Context of the evaluation step in the ECo process

The manufacturing scenario corresponds to the one presented in the introduction of this case study in Section 2.1.1. All resources are available at all the time, no breakdowns or stochastic production times are assumed. Additionally, for each product the following dimensions are assumed as shown in Table 5.1. For all products it is assumed that they can be rotated in all three dimensions.

There exists one type of loading device that can be used. The loading device has the following dimensions: $20 \times 15 \times 15$ (height, width, depth) and per loading device costs of 25 units have to be paid. A loading device can store items up too 20 units of weight. For packaging there is place to use 5 different loading devices in parallel.

job type	weight	height	width	depth
J1	2	10	10	10
J2	1	8	6	4
J3	3	20	15	6
J4	3	10	10	8
J5	1	5	6	8

Table 5.1.: Dimensions and weight for products

For the transportation planning a two dimensional Cartesian coordinate system is assumed with a space of 200×200 coordinates, with coordinates ranging from $(-100, -100)$ to $(100, 100)$. The depot is in the center at the point $(0, 0)$. The distance between two points is measured by the Euclidian distance.

Each truck has the capacity to transport only one loading device. Per time unit it can travel a distance of 4 units. For each distance unit a cost of one unit is incurred for traveling. For the unloading of goods at the customer site, one time unit is necessary. In total there are three trucks available for distribution.

This information is kept constant in all experiments performed in this study. We changed the number of orders and their content. An order has specifies the required product, its quantity, the coordinates of the customer and its due date. This data is generated randomly. The product is chosen randomly out of all five products. The number of required products is chosen randomly from one up to five products per order. The coordinates (x and y coordinate) of the customer are also chosen randomly. The computation of the due date of an order is more complex. A minimal duration x is computed as follows:

$$\mathsf{x} = 3 * \mathit{minimal\_production\_time} + \mathit{travel\_time} + 96$$

The due date is chosen randomly from the interval $[x, 2x]$. If a due date is missed for an order, costs of 25 units are incurred per time unit of delay. Of course, the number of orders that have to be processed is another degree of freedom that is varied during the experiments.

During the tests performed for this study the results of the local planning systems are not deterministic. To describe our experimental settings we use the following terminology. A replication of a test is the repeated execution of a test, based on identical input data. *Replications* are necessary because the used planning systems do not generate deterministic results. That is, the computed plans can vary in different test runs with identical input data.

An *instance* describes a test instance, with a fixed number of orders. The degree of freedom, and thus, different complexity for the production system results from the product mix and the quantities that have to be produced for each order. Both product mix and requested quantities per order can change per instance.

A *scenario* is characterized by a fixed number of orders. Consequently, there different instances can be generated for one scenario.

We present a comparison between the two coordination mechanisms that have been discussed before. First, we investigate the performance of both approaches under different loads, i.e., different number of orders. Thereafter, we investigate in more depth the performance concerning different instances for three different scenarios.

The results discussed in the following are based on 30 different scenarios. For each scenario 10 different instances have been created, and one replication is done for each instance. For each scenario, the mean costs have been computed based on the results of the 10 different instances. To give a better overview of the results, we present here the results split into three different figures, instead of one, as the differences between the smaller instances cannot be identified in the overall figure.

Note that by simply increasing the number of orders, the stress of the system is likely to increase, and therefore, the induced costs. This is not a necessity and not in every time this is true. Therefore we have identified two reasons. First, the complexity per order can change, as the products and the required quantity can differ per order. This variety effects the performance of the production system. To reduce these effects, different instances of the same scenario are tested. Second, the underlying decision problems are planning and scheduling problems. These problems do not have a have a strict linear increase in complexity.

For the first 10 scenarios (1 to 10 orders) the results of the sequential and the improved coordination approaches are depicted in Figure 5.9.

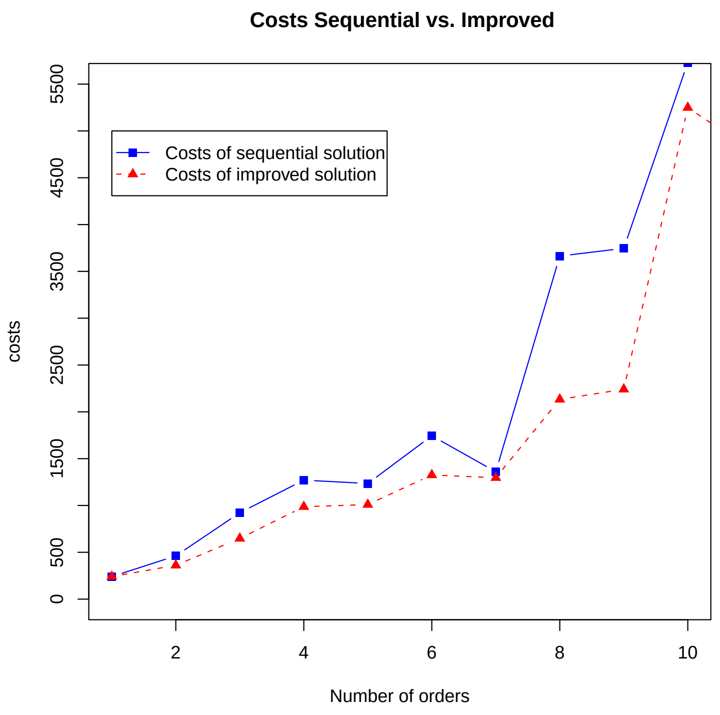


Figure 5.9.: Performance comparison, part 1

The second part of the scenarios (10–20) is shown in Figure 5.10. While in the first part of this experiment penalties are rare and costs are caused mainly by transportation, in the second part most of the costs are caused by missed dues, which is a clear indication that the production system comes from a normal operational state to an overload situation.

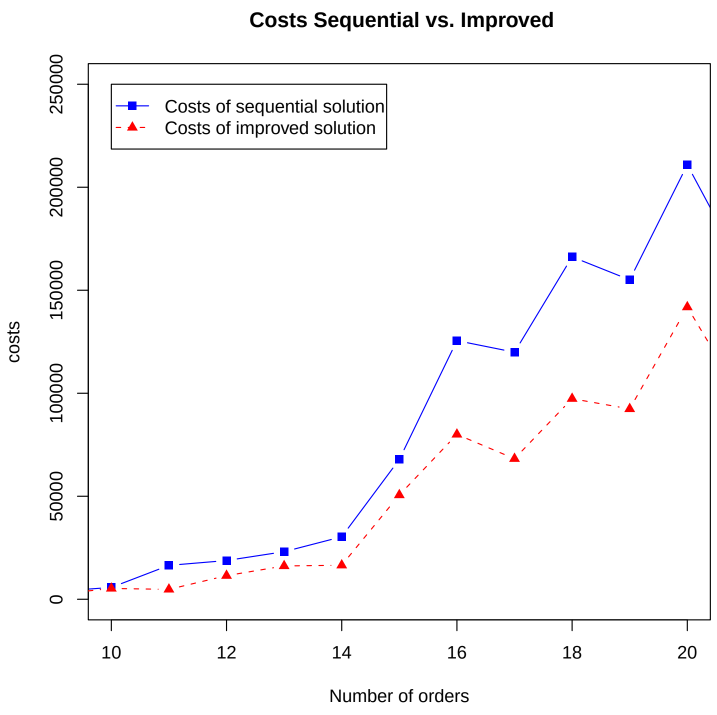


Figure 5.10.: Performance comparison, part 2

Finally, in the third part (scenarios with 20–30 orders) costs are tremendous. The graph is shown in Figure 5.11. This clearly indicates that the load of the production system is far too high. During all scenarios the mean costs of the improved coordination mechanism are below the costs of the sequential coordination of the planning systems. But due to the limited number of instances and replications, this can only be stated as a first observation. This data is not reliable to draw conclusions from.

We now investigate in more depth three different scenarios to compare the coordination approach in contrast to each other. The first scenario is from the first part, where typically the orders can be satisfied without penalties. Therefore, we choose a load of 5 orders. The second scenario is one in the critical section, where instances exist that do not cause penalties and others where large amounts

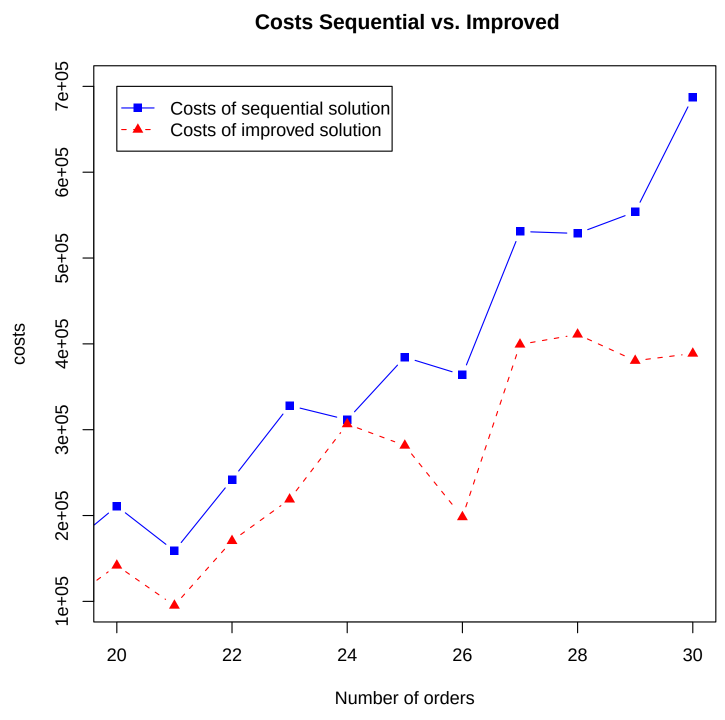
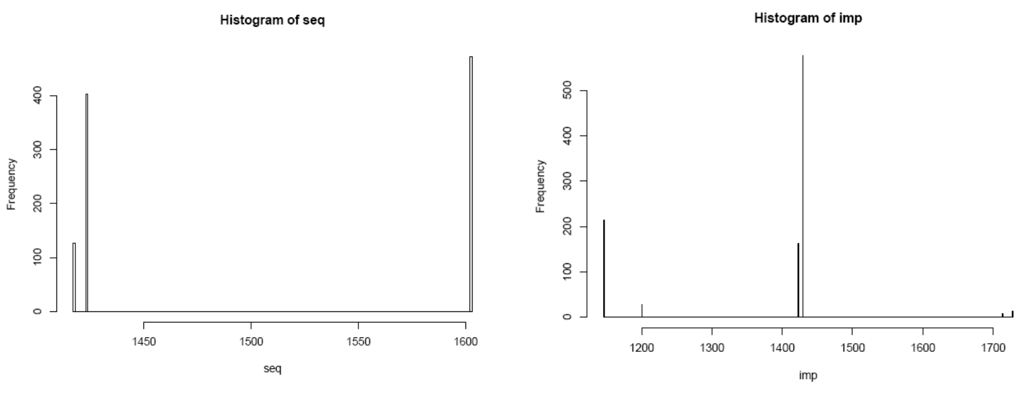


Figure 5.11.: Performance comparison, part 3

of penalties have to be paid. Therefore, we present the results of a scenario with 10 orders. Finally, we instigate a scenario with 15 orders, so a situation where a clear overload of the system exists. For this more in depth investigation we use 10 instances per scenario and for each instance we compute 1000 replications. We use an analysis per instance here, as it does not seem appropriate for a more detailed analysis to compare the results of the performance for different sets of orders. Consequently, we have performed 30 analysis different that lay the foundation for this discussion. All results that could be obtained are summarized in the Appendix B.4.

First, even though in most instances the results of the 1000 replications differs (there was only one instance that leads to the same results in all 1000 replications of the sequential and the improved results) it can be stated that in most instances the number of different solutions is quite limited. For the instances (1–10), for the first scenario with 5 orders, the mean number of different results is about 7. The minimal number of solutions for one instance is 1 and the maximal number of different solutions for the same input is 20. An exemplary distribution of solutions is shown in Figure 5.12 where the histogram of the instance number nine is shown



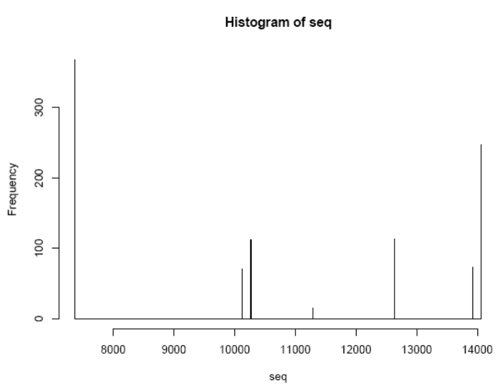
(a) sequential (b) improved

Figure 5.12.: Histogram for the sequential and improved approach for instance 9

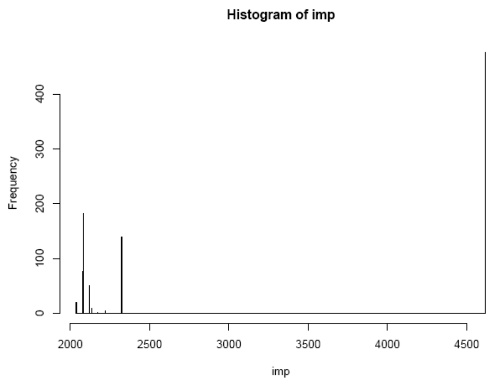
that leads to three different solutions using the sequential planning approach and 6 different solutions using the improved planning approach. Note that the improved approach does not always lead to a more diverse solutions space. For the first scenario the mean number of solutions with the sequential solutions is 6.8 while for the improved it is 7.9.

For the second and third scenarios (10 and 15 orders) the number of different solutions increases to 19.1 and 39.1 for the in mean values. It can be assumed that a higher load might lead to greater plan diversity, as more options have to be regarded during the planning. In fact this is not true for all larger instances. There exist instances like (instance 14 or instance 29) that have only a few different solutions. The difference between the diversity between the sequential and the improved planning approach seem not to be correlated to the problem size, even though the data collection here is to weak to allow a final judgement. The mean number of solutions for the sequential approach are 10.2 and 39 for the scenario two and three. The diversity of the improved approach increases in scenario two to 28 and is only slightly above the number of solutions of the sequential solution for scenario three (from 39 to 39.2). Exemplified histograms can be seen in Figure 5.13 for the second and Figure 5.14 for the third scenario.

As it becomes clear from the analysis of the histograms, there are only relatively few different solutions that are computed. Therefore, in addition to the mean value the mode has been analyzed, too. The mode of a data set is the value that has the largest number of occurrence in the data set. That is, it is the highest peak in the histogram. Thus, the different mechanisms are compared according to their resulting mean, their mode value, and their standard deviation.

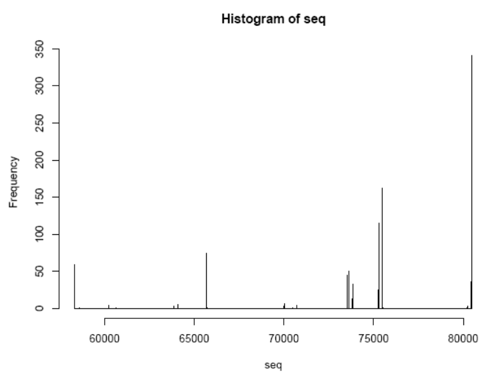


(a) sequential

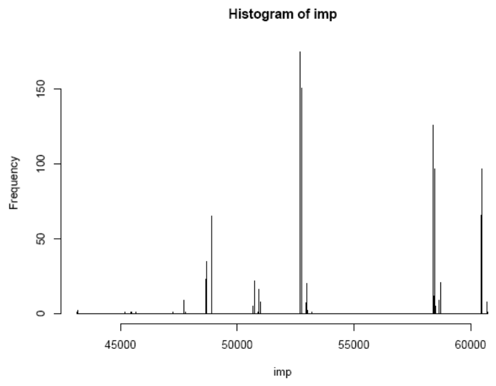


(b) improved

Figure 5.13.: Histogram for the sequential and improved approach for instance 17

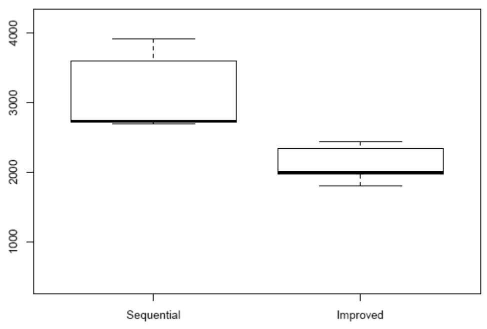


(a) sequential

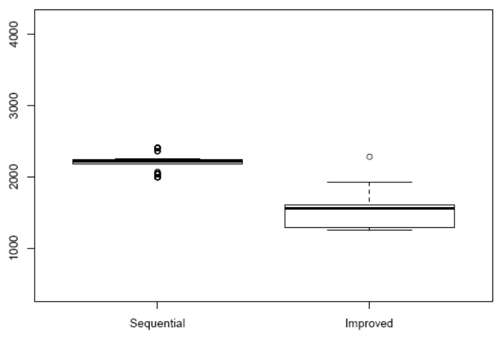


(b) improved

Figure 5.14.: Histogram for the sequential and improved approach for instance 24



(a) Instance 2



(b) Instance 5

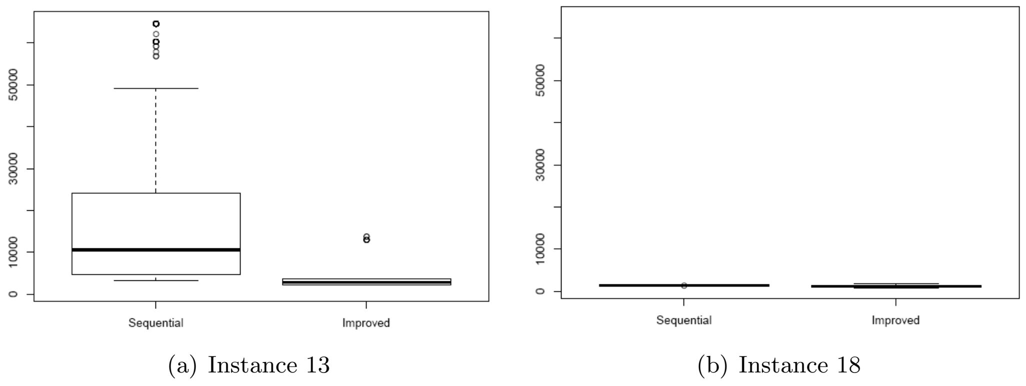
Figure 5.15.: Box plots for instance 2 and 5 comparing sequential and improved coordination approach

All results of this comparison are presented in the Appendix B.4. It could be shown that in 29 out of 30 instances the improved mechanism could outperform the sequential approach measured by the mean and the mode value. Only for the instance 21, the mean and mode value of the improved approach are slightly worse than the sequential approach. Thereby, it has to be noted that this scenario is the only one where no statistically significant difference could be shown between both approaches (p-value=0.9125). For all other instances the pairwise t-test shows high significance results. For an confidence rate of 95% the p-values is always below 10^{-e8}.

Concerning the statical spread, the judgement is not that clear. In 18 out of 30 instances the spread could be reduced with the improved approach, but in 11 out of 30 instances the spread is greater then the sequential one. In instance 6 the spread is equal (0.0), as all results for the sequential and the improved approach do not differ, even though the improved approach performs better then the sequential one.

In the following we discuss some selected results taken out of all three scenarios. All result of this experiments are summarized for all instances in Appendix B.4. From the first scenario the instances 2 and 5 are discussed. Their corresponding box plots based on the 1000 replications done for these instances are shown in Figure 5.15. In these scenarios most orders are performed within their due date, so resulting costs are caused by transportation and packing.

In both instances, as well as in all other instances of this scenario, the mean costs of the improved approach are smaller than in comparison to the costs induced by the sequential approach. Additionally, in instance 2 the spread could be reduced by the improved coordination approach. This could be observed in 5 of 10 instances



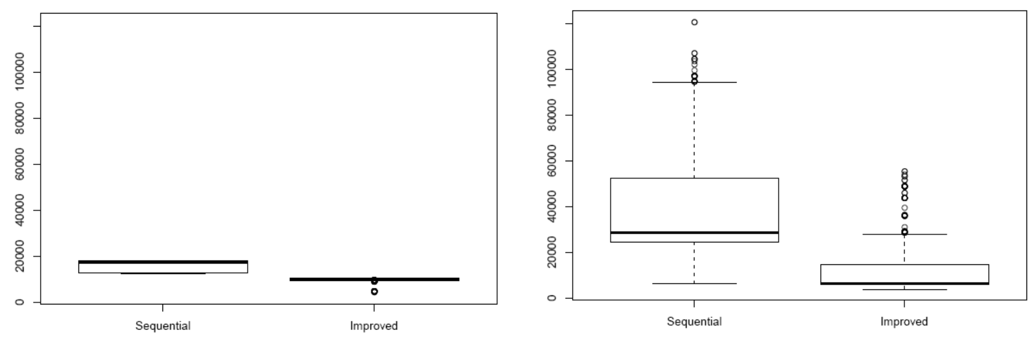
(a) Instance 13 (b) Instance 18

Figure 5.16.: Box plots for instance 13 and 18 comparing sequential and improved coordination approach

of the first scenario. In one instance (instance 6) the spread is equal for both approaches as there exists no spread, at all. In the remaining four instances the spread of the improved coordination mechanism is greater than for the sequential approach. This includes instance 5, which is shown in Figure 5.15 on the right hand side. In this instance the statistical outlier of the improved coordination approach with costs of 2284 units is quite close to the mode costs of the sequential costs (2226 units).

The second scenario with 10 orders is of particular interest, as the production systems is at the border between normal load to overload. For some instances it is possible to satisfy most or all orders without penalties, while for other instances penalties have to be paid. This is one reason why the costs for different instances, but even within one instance, vary largely. The summarized results of two instances are shown in Figure 5.16.

Instance 13 (on the left hand side of Figure 5.16) is of particular interest, as it shows that even within the same instance, and therefore with identical input data, the system can handle the task without heavy penalties but does not have to. Especially the sequential coordination approach has a large spread. In fact, this instance has the largest spread of all instances of this scenario. In this instance the system is at the borderline from a load that could be handled to a clear overload situation, as the costs depend on smaller deviations of sub-plans. In this, as well as five other instances, the spread of the improved coordination approach is smaller than for the sequential approach. In the remaining four instances the sequential approach leads to a smaller spread of costs, even though for all instances the mean costs are higher using the sequential coordination. The coordination with the improved approach can avoid of the penalties in instance 13, its mean costs



(a) Instance 23 (b) Instance 25

Figure 5.17.: Box plots for instance 23 and 25 comparing sequential and improved coordination approach

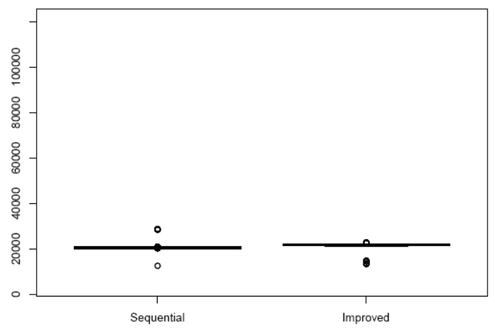
are remarkably lower than the sequential ones.

A scenario where all 10 orders could be performed without any penalty is instance 18, which is shown on the right hand side of Figure 5.16. In this scenario the costs are (1437 and 1265) even lower than the results of instance 2 or 5 discussed above. This deviations can be explained by the fact that even if the number of orders are fixed the load of the system can vary because the number of required products per order is not fixed and the location of customers are picked randomly and thus result in different costs for distribution.

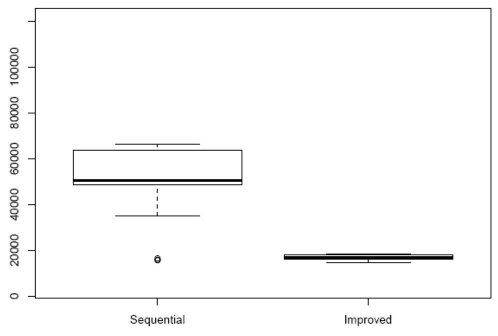
In the third scenario, the state of the production system is in a clear overload situation. Most of the costs are due to penalties that have to be paid. This can be clearly seen as we have tripled the number of orders in comparison to the first scenario (from 5 to 15 orders) while the costs grows more the 25 times. Only in instances 23 and 25 the improved coordination approach is able to generate acceptable results. The results for these two instances are shown in Figure 5.17.

In both approaches, the improved coordination approach is capable of reducing the amount of penalties drastically. Even though in instance 25 the spread is quite large for both approaches the improved approach can generate the best results for this instances. The best solution (with minimal costs) found by the improved approach has only costs of 3938 units, while the mode, i.e., the solution found most often has costs of 6456. In contrast to the sequential mechanism were the best solution found during all replications has costs of 6304 units and the solution found most often has costs of 34444.

In Figure 5.18 the results of the experiments for the instances 21 and 30 are shown. Instance 21 (left hand side of Figure 5.18) is of particular interest, as it is the only instance where the sequential coordination approach performs better



(a) Instance 21



(b) Instance 30

Figure 5.18.: Box plots for instance 21 and 30 comparing sequential and improved coordination approach

than the improved one. The possible reasons for the poor performance of the improved approach are outlined in the following. In the improved approach, the time a products is released for packing is delayed to facilitate synergies between different orders during packing and transportation. Thereby, the time needed for transporting the goods is actually not regarded. Thus, we assume that in this scenario the time for transporting the goods becomes critical, as this extra time leads to the fact that some orders generate higher penalties in contrast to the sequential solution. Even though it has to be mentioned that the difference between both approaches is quite small. In fact, for this instance, the results are quite similar, which is also indicated by the not significant difference between both results in the pairwise t-test. As pointed out above this is the only instance where the t-test could not show significant differences between both approaches, with a very high p-value (p-value=0.9125).

Instances 30, in contrast, is a rather typical instance of this scenario. The improved coordination approach offers a drastic reduction of penalties (mean costs for sequential plans are 53174.97, while mean costs for improved plans are 16875.16). In 9 out of 10 instances, the improved coordination approach results in plans with lower costs. Additionally, the statistical spread is lower for the plans generated by the improved approach in instances 30. This effect can be observed in 7 out of 10 instances of this scenario in favor for the improved coordination approach.

5.7. Criticisms of the ECo-CoPS approach

In this chapter we have applied the ECo-CoPS approach on the SPT case study. For that reason we have presented the modeling of the approach in Section 5.2. The formal model turned out to be very useful as no suitable coordination mechanism exists, because the model eases the process of designing and implementing an appropriate coordination mechanism. For that reason, it might be useful to investigate the potential of other modeling techniques that facilitate a formally grounded description and allow usage during implementation. Such modeling techniques can foster a potential implementation phase.

The variety of coordination requirements was not in the focus of this case study. The coordination requirements are mostly functional, i.e., the plans from the different planning systems have to be coordinated and their execution has to minimize induced costs. For that reason we have identified two coordination approach that do fulfill this requirement. These are a sequential coordination approach, and an improved coordination scheme, which has been designed specifically for this situation, based on principles of coordination approaches from negotiation and result sharing.

One point worth mentioning that was striking during the implementation phase, is the adaptation of existing planning systems. The input and output of the existing planning systems is favoring the sequential execution of planning steps, starting with the production followed by packaging and transportation. Even if these interfaces are adapted by web services, it was necessary to develop additional decorators to adapt the planning systems for the coordination agents. Therefore, the design of interfaces for planning systems is an issue that can limit the options of adapting existing planning systems or can at least make this step harder.

In the evaluation it turned out that the improved coordination approach can generate more efficient coordinated plans in contrast to the sequential planning process which was initially proposed by the students.

6. Engineering Coordination: The SCM case study

In this chapter we discuss the application of the ECo-CoPS approach in a production network and the selection of an existing coordination mechanism for the particular coordination requirements that exist in production networks. For the validation done in this chapter we have reduced the complexity of the scenario, primarily to reduce computational effort. In this case study we relate to previous work done by the author in the field of coordination within supply chains, in particular we refer to [Sch04, SS05, SLT08a].

6.1. The SCM case study

This chapter is based on the case study presented in Section 2.1.2. We apply the ECo-CoPS approach here. Therefore, we investigate a scenario that has been designed to allow an analytical evaluation. For that reason we have introduced a scenario with two companies and two products offered by the network. This information is summarized in Table 6.1 and Table 6.2[1].

Product	Variant	Operations
1	11	111, 112, 113, 114
2	21	211, 212, 213, 214

Table 6.1.: Product descriptions of the production network

company	offered operations
A	111, 114, 211, 212, 213, 214
B	112, 113, 211, 212, 213, 214

Table 6.2.: Capabilities of production network members

[1]These tables restate the scenario described in Section 2.1.2.

While the production of A requires a purely collaborative behavior. The required capabilities for the second product are distributed among both companies, that they have overlapping competences. This adds competitive elements to the relation between both companies. Within this case study we point out different non-functional requirements that are specific for production networks in which members have to cooperate but have overlapping competences. In the following we describe the five steps of the ECo process in detail for this case study.

6.2. Modeling the scenario

At first, we describe the modeling step of the ECo process. The context of this process step with the overall ECo process is shown in Figure 6.1.

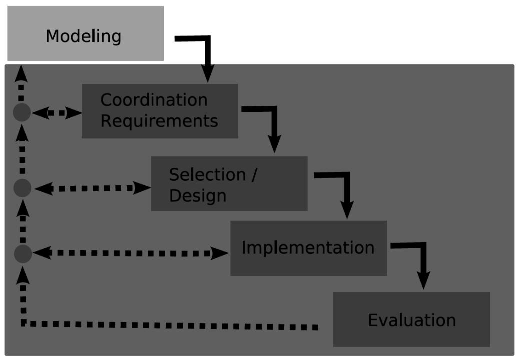


Figure 6.1.: Context of the modeling step in the ECo process

The modeling of the entire planning process within a supply chain can become a complex and elaborated task. We assume some simplification here, for instance, it is assumed that each entity within the supply chain has to solve a scheduling problem, internally. Transportation tasks are not dealt with explicitly. It is assumed that transportation tasks are planned by the supplying entity. As we have already presented a model for a scheduling problem in the previous chapter, we omit this part of the model in this section and focus on the network wide aspects. Note that we discuss a generalization of supply chain here. We use in this case study the concept of production network, which is a generalization of supply chains in the production domain, see Corsten and Gössinger [CG01]. Other forms of production networks that have been discussed in the literature are virtual enterprises, see Corsten and Gössinger [CG01] and Goranson [Gor99, pp. 65–68].

Before we are going into details of modeling, we give an overview of the scenario that is modeled here. The resulting two tier scheduling problem is sketched in Figure 6.2. It is modeled in form of an AND/OR tree. The model is inspired by the production scheduling model presented by Sauer [Sau93], that has been extended towards a two tier concept in subsequent work of this author [Sau02]. The model presented here is extended in the way that is does not address primarily the planning problem but focus more on the representation of planning author-

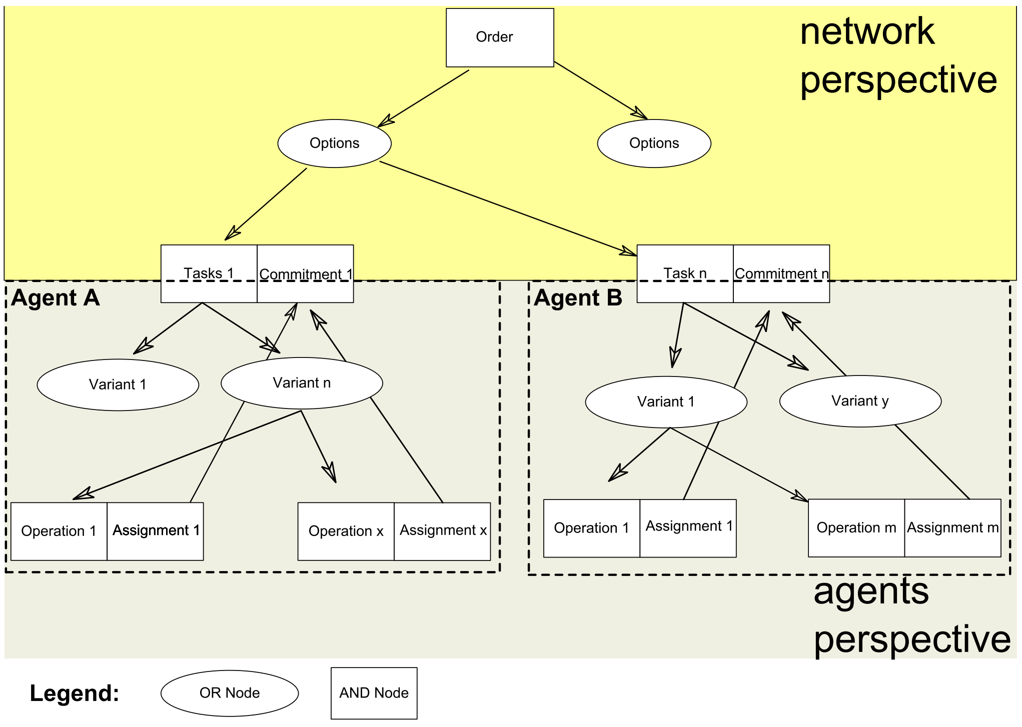


Figure 6.2.: AND/OR tree representing the planning problem

ities by agents and the means that are necessary for modeling the coordination requirements.

As introduced in the previous chapter, we use the same conventions for the formal presentation. Moreover, all concepts, sets and functions defined in this section have been summarized in the Appendix C.1.

We describe a dynamic system here, i.e., the state of the system can changes over time. Time is represented as a discrete increasing integer value. Let $\mathtt{t}$ be a point in time (*pit*) with $\mathtt{t} \in \mathbb{N}$.

A planning entity is represented by a PAA a that act on the behalf of its planning entity. Additionally, a CA c is used to represent a centralized data collection point, that offers services to the agents belonging to the network. To refer to all PAAs and CAs we introduce the following sets.

Set Definition 6.2.1

The set of all CAs is referred to as $\mathcal{C}$.

The set of all PAAs is referred to as $\mathcal{A}$.

Each PAA may be part of more than one network at the same time. Therefore, we have to distinct between the networks. Note that the coordination is done on the network level. We do not intended to establish a coordination mechanism that aims at a coordination over different networks. Even if agents participate in different networks, they have to coordinate their activities within all of their networks separately.

Concept Definition 6.2.1

A **network** ν is defined as $\nu = \langle \mathcal{A}_{\nu,\mathtt{t}}, c\rangle$ with

- $\mathcal{A}_{\nu,t} \subset \mathcal{A}$, the members of the network
- c a coordination agent

As we model a dynamic system the number of members and the composition of the network can change over time. Therefore, the set of members has to be specified for a *pit* $\mathtt{t}$. To refer to all networks we use the set $\mathcal{N}$.

Set Definition 6.2.2

The set of networks is labeled with $\mathcal{N}$.

The attributes of a network can be referred to by projection functions. An agent a is **member** in a network ν at *pit* $\mathtt{t}$, if it is part of a network. This is encoded in the predicate *is_member*.

Function Defintion 6.2.1

The predicate is defined as: $\mathit{is\_member} : \mathcal{A} \times \mathcal{N}_t \rightarrow \{\mathit{true}, \mathit{false}\}$. For a and $nu = \langle \mathcal{A}_{\nu,\mathtt{t}}, c\rangle$ and the current *pit* $\mathtt{t}$ it is defined as:
$\mathit{is\_member}(a, \nu) \Leftrightarrow a \in \mathcal{A}_{\nu,\mathtt{t}}$.

The function $n\_c : \mathcal{N} \rightarrow \mathcal{C}$ for $\nu = \langle \mathcal{A}_{\nu,\mathtt{t}}, c\rangle$ is defined as: $n\_c(\nu) = c$.

As the composition of the network can change over time, the result of this function depends on the state of the set of member at the time of execution.

For a network ν the CA is **unique**, i.e., for all networks ν and μ the following condition holds:

$$\forall \nu, \mu \in \mathcal{N}, \nu \neq \mu \Rightarrow n\_c(\nu) \neq n\_c(\mu)$$

A network $\nu = \langle \mathcal{A}_{\nu,\mathtt{t}}, c\rangle$ is **active**, if it has at least one PAA and a CA. This is encoded in the predicate *is_active*.

Function Defintion 6.2.2

A network $\nu = \langle \mathcal{A}_{\nu,\mathtt{t}}, c\rangle$ is active $\mathit{n\_active} : \mathcal{N} \rightarrow \{\mathit{true}, \mathit{false}\}$ with $\mathit{n\_active}(\nu) \Leftrightarrow \mathcal{A}_{\nu,\mathtt{t}} \neq \emptyset \wedge c \neq \emptyset$

For the rest of this document, we assume that all networks are active.

Each agent offers a set of capabilities to a network. Capabilities are modeled only by their identifiers[2] here.

Set Definition 6.2.3

Let $\mathcal{B}$ be a formal language over Σ used to label the **capabilities**.

Let $\mathcal{B}$ be mutually unambiguous towards the other presented sets of labels. Each agent has a set of capabilities, the **capabilities of the agent**.

Set Definition 6.2.4

The capabilities of agent a are referred to as $\mathcal{B}_a \subseteq \mathcal{B}$.

At any *pit* $\mathtt{t}$ an agent a can use a subset of those capabilities, which are encoded as $\mathcal{B}_{a,\mathtt{t}}$. The capabilities that an agent has at a given point of time are given by the function b_has_a.

Function Defintion 6.2.3

This function is defined as: $b\_has_a : \wp(\mathcal{B}) \times \mathbb{N} \rightarrow \wp(\mathcal{B})$. For a at *pit* $\mathtt{t}$ it is defined as $b\_has_a(\mathcal{B}_a, \mathtt{t}) = \mathcal{B}_{a,\mathtt{t}} \subset \mathcal{B}_a$.

This function is defined by the planning authority of the agent and can be subject of change. A subset of these capabilities is offered to each network the agents is participating in.

Set Definition 6.2.5

The set of **offered capabilities by the agent** is referred to as $\mathcal{B}_{a,\mathtt{t},\nu}$.

This set is specified by the planning authority using the function b_allow_a.

Function Defintion 6.2.4

The function is defined as: $b\_allow_a : \wp(\mathcal{B}_a) \times \mathcal{N} \times \mathbb{N} \rightarrow \wp(\mathcal{B}_a)$. For a at *pit* $\mathtt{t}$ with $\mathcal{B}_{a,\mathtt{t}}$ and ν it is defined as $b\_allow_a(\mathcal{B}_{a,\mathtt{t}}, \nu, \mathtt{t}) = \mathcal{B}_{a,\mathtt{t},\nu}$.

Set Definition 6.2.6

The **capabilities of a network** $\nu = \langle \mathcal{A}_{\nu,\mathtt{t}}, c \rangle$ at *pit* $\mathtt{t}$ can be derived as

$$\mathcal{B}_{\nu,\mathtt{t}} = \bigcup_{\forall a \in \mathcal{A}_{\nu,\mathtt{t}}} \mathcal{B}_{a,\mathtt{t},\nu} :$$

The simplest executable steps on the network level are **jobs**. For the execution of a job a set of capabilities is required.

[2]More sophisticated models for capabilities have been proposed by Lorenzen et al. [LWD+06] or Timm et al. [TSH06], for instance.

Concept Definition 6.2.2

A job is defined as $j = \langle \mathcal{B}_j \rangle$, with $\mathcal{B}_j \subset \mathcal{B}$

A job specifies a set of capabilities that are required to execute it. The set of jobs known to a network are all jobs that have been defined within the network.

Set Definition 6.2.7

Let $\mathcal{J}_\nu$ be the set of **jobs of network** ν.

The **required capabilities for job** j can be retrieved by the function j_b.

Function Defintion 6.2.5

The function j_b is defined as $j\_b : \mathcal{J}_\nu \rightarrow \wp(\mathcal{B})$. For $j = \langle \mathcal{B}_j \rangle$ it is defined as $j\_b(j) = \mathcal{B}_j$.

A job can be performed by any agent within the network that offers all required capabilities to this network. More formally, agent a a member of network ν can execute job j, a job of network ν, iff $j\_b(j) \subseteq \mathcal{B}_{a,\mathrm{t},\nu}$.

From the perspective of each local planning entity those jobs are the orders that have to be performed with their local resources. This typically forms a scheduling problem for all activities that have to be performed on the local level. We do not consider the local scheduling activities and do not model them here explicitly. This can be done analogous to the modeling of the scheduling problem described in Section 5.2. Of course, we have to assume that the local planning entities generate only valid/feasible schedules and that its coordination activities are based on such consistent plans.

To perform some meaningful services or products on the network layer a set of jobs have to be performed in some sequence. This can be encoded in form of a variant.

Concept Definition 6.2.3

A **variant** v is defined as $v = \langle \mathcal{J}_v, <_v \rangle$, with $\mathcal{J}_v \subset \mathcal{J}_\nu$ a set of jobs and $<_v$ an infix operator encoding a partial order among $\mathcal{J}_v$.

With the partial order $<_v$ the sequencing of different jobs can be encoded. If two jobs $j_1, j_2 \in \mathcal{J}_v$ with $j_1 <_v j_2$ it implies that job j_1 has to be finished before job j_2 can be started.

Set Definition 6.2.8

The **set of all variants** known to a network ν is referred to $\mathcal{V}_\nu$.

A variant can be seen as a directed graph. Representing the jobs as the nodes and the ordering relation $<_v$ defines the edges in the graph.

Set Definition 6.2.9

The **set of all predecessor** of a job in a variant is defined as $\mathcal{J}_{pre,v,j}$.

The **set of all successors** of a job in a variant is defined as $\mathcal{J}_{suc,v,j}$.

The set of all predecessor $\mathcal{J}_{pre,v,j}$ and successors $\mathcal{J}_{suc,v,j}$ can be computed by graph search algorithms, see Sedgewick [Sed98, pp. 537–541].

The functions j_pre and j_suc compute the set of all predecessors and successors in the graph formed by v for job j.

Function Defintion 6.2.6

The function j_pre is defined as $j\_pre : \mathcal{V}_\nu \times \mathcal{J}_\nu \to \wp(\mathcal{J}_\nu)$. For v and $j \in \mathcal{J}_v$ it is defined as $j\_pre(v, j) = \mathcal{J}_{pre,v,j}$.

The function j_suc is defined as $j\_suc : \mathcal{V}_\nu \times \mathcal{J}_\nu \to \wp(\mathcal{J}_\nu)$. For v and $j \in \mathcal{J}_v$ it is defined as $j\_suc(v, j) = \mathcal{J}_{suc,v,j}$.

The function v_j is defined as $v\_j : \mathcal{V}_\nu \to \wp(\mathcal{J}_\nu)$. For $v = \langle \mathcal{J}_v, <_v \rangle$ it is defined as $v\_j(v) = \mathcal{J}_v$.

A variant specifies production knowledge that is available to the network. This knowledge captures information how jobs can be combined to build **products**.

Concept Definition 6.2.4

A product is defined as $p = \langle \mathcal{V}_p \rangle$, with $\mathcal{V}_p \subset \mathcal{V}_\nu$ a set of variants.

It is possible that different variants are defined for a product. A similar assumption is used by Sauer [Sau93, pp. 25,26].

The knowledge of how to produce a set of products is an asset of the network that enables the network to offer those products to the market.

Set Definition 6.2.10

The set of products of a network ν is referred to as $\mathcal{P}_\nu$.

The variants of a product can be referred to be the function p_var.

Function Defintion 6.2.7

The function p_var is defined as $p\_var : \mathcal{P}_\nu \to \wp(\mathcal{V})$. For $p = \langle \mathcal{V}_p \rangle$ it is defined as $p\_var(p) = \mathcal{V}_p$.

Based on the concept of products, we can define the orders of a network. An **order of a network** o is a request, that the network has to fulfill.

Concept Definition 6.2.5

An order is defined as $o = \langle \nu, p, a, \mathsf{e}, \mathsf{d} \rangle$.
The elements are defined as

- $\nu \in \mathcal{N}$ the network
- $p \in \mathcal{P}_\nu$ the product
- $a \in \mathcal{A}$, $is\_member(a, n, \mathsf{t})$ the representative of this order
- $\mathsf{e}, \mathsf{d} \in \mathbb{N}, \mathsf{d} > \mathsf{e}$ time interval for the fulfillment

The idea of a representative for each order is inspired by concepts of *virtual enterprises*. Virtual enterprises are presented, e.g., by Arnoldt et al. [AFHS95] and Goranson [Gor99]. The representative is the entity that is visible to the customer and gets evaluated by him according to the fulfillment of the order. Therefore, the representative is interested in the correct execution of the order on time. This modeling approach contrasts to more conventional modeling approaches, where orders and resources are both modeled as agents, like, for instance, in Schumann and Sauer [SS07]. Hereby, we emphasize that an order is not a first class entity, as it is processed and not active at all. It is a task agents have to fulfill.

Set Definition 6.2.11

The **set of all orders** is referred to as $\mathcal{O}$.

We define a number of projective functions here that enable us to specify particular elements of an order easily.

Function Defintion 6.2.8

For order $o = \langle \nu, p, a, \mathsf{e}, \mathsf{d} \rangle$ we define the following functions.

The function o_ν is defined as $o\_\nu : \mathcal{O} \to \mathcal{N}$ it is defined as: $o\_\nu(o) = \nu$.

The function o_rep is defined as $o\_rep : \mathcal{O} \to \mathcal{A}$ it is defined as $o\_rep(o) = a$.

The function o_pro is defined as $o\_pro : \mathcal{O} \to \mathcal{P}_\nu$ it is defined as $o\_pro(o) = p$.

The function o_start is defined as $o\_start : \mathcal{O} \to \mathbb{N}$ it is defined as $o\_start(o) = \mathsf{e}$.

The function o_end is defined as $o\_due : \mathcal{O} \to \mathbb{N}$. it is defined as $o\_end(o) = \mathsf{d}$.

An order o is published at *pit* t, this is done by publishing it to the representative of the order. When it has been published it is called **active**. This is encoded in the predicate o_active.

Function Defintion 6.2.9

The predicate o_active is defined as: $d\_active : \mathcal{D} \to \{true, false\}$. For $o = \langle n, a, p, \mathsf{e}, \mathsf{d} \rangle$ it is defined as: $o\_active(o) \Leftrightarrow a \neq \emptyset$

In the following we assume that for all orders are active.

Orders belong to a network. Thus, we can define the **orders of the network** ν.

Set Definition 6.2.12

For the network $\nu = \langle \mathcal{A}_{n,\mathsf{t}}, c \rangle$ at *pit* t the orders of the network are referred to as $\mathcal{O}_{\nu,\mathsf{t}}$. Formally this set is build as follows:

$$o \in \mathcal{O}_{\nu,\mathsf{t}} \Leftrightarrow o \in \mathcal{O} \wedge o\_active(o) \wedge o\_\nu(o) = \nu$$

Each agent can be a representative for orders and can be a member in more than on network at the same time. Therefore, it is worth to define the **orders of the agent** a at *pit* t.

Set Definition 6.2.13

The set of orders of an agent are referred to as $\mathcal{O}_{a,\mathsf{t}}$. This set is defined as follows:

$$o \in \mathcal{O}_{a,\mathsf{t}} \Leftrightarrow o \in \mathcal{O} \wedge o\_active(o) \wedge o\_rep(o) = a$$

If an order has been given to a network it must be processed. The jobs that have been specified in a variant of this product have to be executed. Therefore, it is necessary to decide which variant of the product has to be executed. This is encoded in the function *o_ref*.

Function Defintion 6.2.10

The function *o_ref* is defined as $o\_ref : \mathcal{O} \rightarrow \mathcal{V}_\nu$. For $o = \langle \nu, p, a, \mathsf{e}, \mathsf{d} \rangle$ it is defined as $o\_ref(o) = v$, with $v \in p\_var(o\_pro(o))$.

We do not go into detail how the variant is chosen from the set of existing variants.

An **instantiated job** j^i is a job j, that is part of the variant $v = \langle \mathcal{J}_v, <_v \rangle$. that is necessary for the execution of order o. An initiated job is that fraction of an order that is distributed among the agents of the network. Thus, instantiated jobs are the local orders of the agents.

Concept Definition 6.2.6

An instantiated job is defined as: $j^i = \langle o, \mathcal{B}, \mathcal{J}_1, \mathcal{J}_2 \mathsf{e}, \mathsf{d} \rangle$, with

- o the order that the instantiated job belongs to
- $\mathcal{B} = j\_b(j)$ the required capabilities
- $\mathcal{J}_1 = j\_pre(v, j)$ the jobs j^i depends on
- $\mathcal{J}_2 = j\_suc(v, j)$ the jobs depend on j^i
- $\mathsf{e} \geq o\_start(o)$ start of the execution interval

- $\mathsf{d} < o\_end(o)$ suggested end of the execution interval

The instantiated job $j^i = \langle o, \mathcal{B}, \mathcal{J}_1, \mathcal{J}_2\mathsf{e}, \mathsf{d}\rangle$ **depends** on the jobs in $\mathcal{J}_1$. If $\mathcal{J}_1$ is empty j^i is called **independent**.

Set Definition 6.2.14

The set of all **instantiated jobs of an agent** a at *pit* t is referred to as $\mathcal{J}^i_{a,\mathsf{t}}$.

The set of all **instantiated jobs of a network** ν at *pit* t is referred to as $\mathcal{J}^i_{\nu,\mathsf{t}}$.

A job j is transformed into an instantiated job j^i by the **instantiate job function**.

Function Defintion 6.2.11

This function is defined as $j\_inst : \mathcal{J}_\nu \times \mathcal{O}_{\nu,\mathsf{t}} \rightarrow \mathcal{J}^i_{\nu,\mathsf{t}}$. For $o = \langle \nu, p, a, \mathsf{e}, \mathsf{d}\rangle$ and $j = \langle \mathcal{B}_j \rangle$ with $j \in o\_ref(o)$ it is defined as:

$$j\_inst(j, o) = j^i = \\ \langle o, j\_b(j), j\_pre(o\_ref(o), j), \\ j\_suc(o\_ref(o), j), d\_start(d), d\_due(d)\rangle$$

For the instantiated job $j^i = \langle o, \mathcal{B}, \mathcal{J}_1, \mathcal{J}_2\mathsf{e}, \mathsf{d}\rangle$ we define the following functions.

The function j^i_o is defined as $j^i\_o : \mathcal{J}^i_{\nu,\mathsf{t}} \rightarrow \mathcal{O}$ it is defined as $j^i\_o(j^i) = o$.

The function j^i_ν is defined as $j^i\_\nu : \mathcal{J}^i_{\nu,\mathsf{t}} \rightarrow \mathcal{N}$ it is defined as $j^i\_\nu(j^i) = o\_\nu(j^i\_o(j^i))$.

The function j^i_rep is defined as: $j^i\_rep : \mathcal{J}^i_{\nu,\mathsf{t}} \rightarrow \mathcal{A}$ it is defined as $j^i\_rep(j^i) = o\_rep(j^i\_o(j^i))$.

The function j^i_b is defined as: $j^i\_b : \mathcal{J}^i_{\nu,\mathsf{t}} \rightarrow \wp(\mathcal{B})$ it is defined as $j^i\_b(j^i) = \mathcal{B}$.

The function j^i_jd is defined as $j^i\_jd : \mathcal{J}^i_{\nu,\mathsf{t}} \rightarrow \wp(\mathcal{J}_\nu)$ it is defined as $j^i\_jd(j^i) = \mathcal{J}_1$.

The function j^i_js is defined as $j^i\_js : \mathcal{J}^i_{\nu,\mathsf{t}} \rightarrow \wp(\mathcal{J}_\nu)$ it is defined as $j^i\_js(j^i) = \mathcal{J}_2$.

The function j^i_start is defined as $j^i\_start : \mathcal{J}^i_{\nu,\mathsf{t}} \rightarrow \mathbb{N}$ it is defined as $j^i\_start(j^i) = \mathsf{e}$.

The function j^i_due is defined as $j^i\_due : \mathcal{J}^i_{\nu,\mathsf{t}} \rightarrow \mathbb{N}$ it is defined as $j^i\_due(j^i) = \mathsf{d}$.

An instantiated job j^i is **allocatable** at an agent a at *pit* t iff the predicate $j^i t\_alc$ is *true*.

Function Defintion 6.2.12

This predicate $j^i t\_alc$ is defined as $j^i t\_alc : \mathcal{J}^i_{\nu,\mathsf{t}} \times \mathcal{A} \times \mathbb{N} \rightarrow \{true, false\}$. For $j^i = \langle o, \mathcal{B}, \mathcal{J}_1, \mathcal{J}_2 \mathsf{e}, \mathsf{d} \rangle$, $a \in \mathcal{A}$ and t the predicate is defined as:

$$j^i t\_alc(j^i, a, \mathsf{t}) \Leftrightarrow is\_member(a, j^i\_\nu(j^i), \mathsf{t}) \wedge \forall b \in j^i\_b(j^i) \Rightarrow b \in \mathcal{B}_{a,\mathsf{t},\nu}$$

The set of agents, that are capable to execute an instantiated job j^i at *pit* t are called the set of **candidates**.

Set Definition 6.2.15

The set of candidates for an instantiated job j^i at *pit* t is referred to as $\mathcal{A}^C_{j^i,\mathsf{t}}$. Formally $\mathcal{A}^C_{j^i,\mathsf{t}}$ is defined as follows:

$$a \in \mathcal{A}^C_{j^i,\mathsf{t}} \Leftrightarrow a \in \mathcal{A} \wedge j^i t\_alc(j^i, a, \mathsf{t})$$

An agent that is a member of the set of candidates is called a **candidate**.

For the execution of an instantiated job j^i a candidate a has to generate a **proposal**.

Concept Definition 6.2.7

A proposal is defined as $r = \langle j^i, a, \nu, \mathsf{e}, \mathsf{d} \rangle$, with

- j^i, the instantiated job the proposal addresses,
- a, the agent offering the execution,
- ν, the network the proposal belongs to, and
- $[\mathsf{e}, \mathsf{d}]$, the time window for execution.

Set Definition 6.2.16

The set of all **proposals for an instantiated job** j^i is referred to as $\mathcal{R}_{j^i,\mathsf{t}}$.

The proposals of an agent are referred to as $\mathcal{R}_{a,\mathsf{t}}$.

For proposals we define a number of projective functions, to ease the access to elements of a proposal.

Function Defintion 6.2.13

For the proposal $r = \langle j^i, a, \nu, \mathsf{e}, \mathsf{d} \rangle$ we define the following functions.

This function is defined as $r\_cand : \mathcal{R}_{j^i,\mathsf{t}} \rightarrow \mathcal{A}$ it is defined as $r\_cand(p) = a$.

The function r_j^i is defined as $r\_j^i : \mathcal{R}_{j^i,\mathsf{t}} \rightarrow \mathcal{J}_\nu$ it is defined as $r\_j^i(p) = j^i$.

The function r_start is defined as $r\_start : \mathcal{R}_{j^i,\mathsf{t}} \rightarrow \mathbb{N}$ it is defined as $r\_start(r) = \mathsf{e}$.

The function r_due is defined as $r\_due : \mathcal{R}_{j^i,\mathsf{t}} \rightarrow \mathbb{N}$ it is defined as $r\_due(r) = \mathsf{d}$.

As pointed out above, we are not going into detail how each PAA generates its local plan. Each agent generates a local schedule that has to be consistent and is optimized towards a local objective function. To generate a proposal an agent has to evaluate the effects an additional jobs would have on its local schedule. Therefore, it has to compute and evaluate a potential schedule. If the potential schedule seems to offer an increase in the agents local utility or at least is not worsening it significantly, the agent is willing to generate a proposal. The degree to which an agent is willing to accept quality decrease is defined by a specific threshold. The offered interval for the execution of an instantiated job depends on the one hand on the planned execution time in the potential plan but on the other hand on more strategic aspects like adding offsets for redundancy and the offered service level to the corresponding network the instantiated job belongs to.

If different proposals have been placed for an instantiated job one of them has to be chosen. The selection of a proposal can either be done by the CA or the representative of the corresponding order. The **selection of a proposal** r for an instantiated job j^i is described by the function r_sel.

Function Defintion 6.2.14

The function r_sel is defined as $r\_sel : \wp(\mathcal{R}_{j^i,\mathsf{t}}) \times \mathcal{O}_{\nu,\mathsf{t}} \rightarrow \mathcal{R}_{j^i,\mathsf{t}}$. It is defined as $r\_sel(\mathcal{R}_{j^i,\mathsf{t}}, o) = r \in \mathcal{R}_{j^i,\mathsf{t}}$.

Thereby, different aspects of the proposal can be evaluated. If a proposal has been selected a binding **commitment** has to be formed. Here a commitment m is an obligation of an agent $a1 \in \mathcal{A}$ (the commitment-presenter) towards another agent $a2 \in \mathcal{A}$ (the commitment-receiver) to execute an instantiated job j^i within the time interval $[\mathsf{e}, \mathsf{d}]$.

Concept Definition 6.2.8

A commitment is defined as $m = \langle j^i, a1, a2, \mathsf{e}, \mathsf{d}\rangle$. If c is granted at *pit* t the following conditions have to be satisfied:

- $j^i \in \mathcal{J}^i_{a1,\mathsf{t}}$
- $is\_member(a1, j^i\_n(j^i)) \wedge is\_member(a2, j^i\_n(j^i)$
- $\mathsf{e}, \mathsf{d} \in \mathbb{N} : \mathsf{e} < \mathsf{d}$

Set Definition 6.2.17

Let $\mathcal{M}$ be the set of all commitments.

We also define a number of projective functions for commitments, as well.

Function Defintion 6.2.15

For the commitment $m = \langle j^i, a1, a2, \mathsf{e}, \mathsf{d}\rangle$ we define the following functions.

The function m_pres is defined as $m\_pres : \mathcal{M} \rightarrow \mathcal{A}$ it is defined as $m\_pres(m) = a1$.

The function m_rec is defined as $m\_rec : \mathcal{M} \rightarrow \mathcal{A}$ it is defined as $m\_rec(m) = a2$.

The function m_j^i is defined as $m\_j^i : \mathcal{M} \rightarrow \mathcal{J}^i_{\nu,\mathsf{t}}$ it is defined as $m\_j^i(m) = j^i$, if $j^i \in \mathcal{J}^i_{a1,\mathsf{t}}$.

The function m_start is defined as $m\_start : \mathcal{M} \rightarrow \mathbb{N}$ it is defined as $m\_start(m) = \mathsf{e}$.

The function m_due is defined as $m\_due : \mathcal{M} \rightarrow \mathbb{N}$ it is defined as $m\_due(m) = \mathsf{d}$.

A **commitment is active**, if it was published by a commitment-presenter towards a receiver. This is encoded in the function m_active.

Function Defintion 6.2.16

The function m_active is defined as $m\_active : \mathcal{M} \rightarrow \{true, false\}$. For $m = \langle j^i, a1, a2, \mathsf{e}, \mathsf{d}\rangle$ it is defined as: $m\_active(m) \Leftrightarrow m\_pres(m) \neq \emptyset \wedge m\_rec(m) \neq \emptyset$

The **commitments of an agent** a that have been presented by this agent at *pit* t are gathered in the set of active commitments.

Set Definition 6.2.18

For the commitment $m = \langle j^i, a1, a2, \mathsf{e}, \mathsf{d}\rangle$ the following sets are defined.
The set of the **commitments of an agent** a at *pit* t is labeled as $\mathcal{M}_{a,\mathsf{t}}$. The commitment m is in $\mathcal{M}_{a,\mathsf{t}}$ if the presenter $a1$ is a. That is encoded as:

$$m \in \mathcal{M}_{a,\mathsf{t}} \Leftrightarrow m \in \mathcal{M} \wedge m\_active(m) \wedge m\_pres(m) = a$$

The commitments of a representative a at *pit* t are referred to as $\mathcal{M}^{\mathcal{R}}{}_{a,\mathsf{t}}$. The commitment m is in $\mathcal{M}^{\mathcal{R}}{}_{a,\mathsf{t}}$ if the receiver $a2$ is a. That is encoded as:

$$m \in \mathcal{M}^{\mathcal{R}}{}_{a,\mathsf{t}} \Leftrightarrow m \in \mathcal{M} \wedge m\_active(m) \wedge m\_rec(m) = a$$

The **commitments of an agent** a **towards network** ν at *pit* t are combined in the set $\mathcal{M}_{a,\nu,\mathsf{t}}$. Formally this set can be described as:

$$\forall m \in \mathcal{M}_{a,\nu,\mathsf{t}} \Leftrightarrow m \in \mathcal{M}_{a,\mathsf{t}} \wedge j^i\_\nu(m\_j^i(m)) = \nu$$

The **commitments of a network** ν, indicated as $\mathcal{M}_{\nu,\mathsf{t}}$, at *pit* t contains commitments for activities of the network. Formally this can be stated for the network $\nu = \langle \mathcal{A}_{\nu,\mathsf{t}}, c\rangle$ at *pit* t as :

$$\mathcal{M}_{\nu,\mathsf{t}} = \bigcup_{\forall a \in \mathcal{A}_{\nu,\mathsf{t}}} \mathcal{M}_{a,\nu,\mathsf{t}}$$

A commitment is given by the agent, that has presented the selected proposal. It gives the commitment to the representing agent of the order for the execution of the instantiated job specified in the proposal. Thus, a proposal can be transformed into a commitment using the function *m_gen*.

Function Defintion 6.2.17

The function *m_gen* is defined as $m\_gen : \mathcal{R}_{j^i,\mathsf{t}} \to \mathcal{M}$. For $r = \langle j^i, a, \nu, \mathsf{e}, \mathsf{d}\rangle$ it is defined as

$$m\_gen(r) = \langle r\_j^i, r\_cand, j^i\_rep(r\_j^i(r)), r\_start(r), r\_due(r)\rangle = m$$

.

To identify the commitment that belongs to a certain instantiated job j^i the function *m_find* can be used. This function identifies the **commitment for an instantiated job** j^i.

Function Defintion 6.2.18

The function *m_find* is defined as $m\_find : \mathcal{J}^i_{\nu,\mathsf{t}} \times \wp(\mathcal{M}_{\nu,\mathsf{t}}) \to \mathcal{M}_{\nu,\mathsf{t}}$. For $j^i = \langle \nu, \mathcal{B}, a, \mathcal{J}_1, \mathcal{J}_2, \mathsf{e}, \mathsf{d}\rangle$ and $\mathcal{M}_{\nu,\mathsf{t}}$ it is defined as: $m\_find(j^i, \mathcal{M}_{\nu,\mathsf{t}}) = m$, with $m \in \mathcal{M}_{\nu,\mathsf{t}} \wedge m\_j^i(m) = j^i$

Set Definition 6.2.19

The corresponding **commitments of an order** o at *pit* t are referred to as $\mathcal{M}_{o,\mathsf{t}}$. This set is defined for the current *pit* t as

$$m \in \mathcal{M}_{o,\mathsf{t}} \Leftrightarrow m \in \mathcal{M} \wedge j^i\_o(m\_j^i(m)) = o$$

The set of commitments $\mathcal{M}_{o,\mathsf{t}}$ for the order o at *pit* t is **complete** if the predicate *m_comp* is *true*.

Function Defintion 6.2.19

The predicate *m_comp* is defined as $m\_comp : \wp(\mathcal{M}) \times \mathcal{O} \rightarrow \{true, false\}$. It is defined as:

$$m\_comp(\mathcal{M}_{o,\mathrm{t}}, o) \Leftrightarrow$$
$$\forall j^i \in \mathcal{J}^i_{\nu,\mathrm{t}} : j^i\_o = o \exists m \in \mathcal{M}_{o,\mathrm{t}} : m\_j^i(m) = j^i.$$

The set of all commitments of a network ν at *pit* $\mathtt{t}$ is **complete** iff

$$\bigwedge_{\forall o \in \mathcal{O}_{\nu,\mathrm{t}}} m\_comp(\mathcal{M}_{o,\mathrm{t}}, o) \tag{6.1}$$

The **beginning and ending of an order** o are defined as follows.

Function Defintion 6.2.20

The function *o_beg* is defined as $o\_beg : \wp(\mathcal{M}) \rightarrow \mathbb{N}$. For o and its *complete* set of commitments $\mathcal{M}_{o,\mathrm{t}}$ it is defined as

$$o\_beg(\mathcal{M}_{o,\mathrm{t}}) = \min_{\forall m \in \mathcal{M}_{o,\mathrm{t}}} \{m\_start(m)\}$$

The function *o_end* is defined as $o\_end : \wp(\mathcal{M}) \rightarrow \mathbb{N}$. For o and its *complete* set of commitments $\mathcal{M}_{o,\mathrm{t}}$ it is defined as

$$o\_end(\mathcal{M}\_o, \mathtt{t}, d) = \max_{\forall m \in \mathcal{M}_{o,\mathrm{t}}} \{m\_due(m)\}$$

The set of commitments $\mathcal{M}\_o, \mathtt{t}$ for an order o at *pit* $\mathtt{t}$ is **consistent** if the predicate *m_con* is *true*.

Function Defintion 6.2.21

The predicate *m_con* is defined as $m\_con : \wp(\mathcal{M}) \times \mathcal{O}_{\nu,\mathrm{t}} \rightarrow \{true, false\}$. For o and $\mathcal{M}_{o,\mathrm{t}}$ it is defined as

$$m\_con(\mathcal{M}_{o,\mathrm{t}}, o) \Leftrightarrow$$
$$\begin{aligned} &m\_comp(\mathcal{M}_{o,\mathrm{t}}) \wedge \\ &o\_beg(\mathcal{M}_{o,\mathrm{t}}) \geq o\_start(o) \wedge \\ &\forall m_i, m_j \in \mathcal{M}_{o,\mathrm{t}} : m\_j^i(m_i) <_v m\_j^i(m_j) \Rightarrow \\ &m\_due(m_i) < m\_start(m_j) \end{aligned}$$

The set of all commitments $\mathcal{M}_{\nu,\mathrm{t}}$ of a network ν at *pit* $\mathtt{t}$ is **consistent** iff

$$\bigwedge_{\forall o \in \mathcal{O}_{\nu,\mathrm{t}}} m\_con(\mathcal{M}_{o,\mathrm{t}}, o) \tag{6.2}$$

If an order o has been scheduled in a consistent way, i.e., $m\_con(\mathcal{M}_{o,\mathsf{t}}, o)$ is *true*, the execution of the order can be **evaluated** regarding the adherence of its due date. This is done using the function o_eval. This function commonly punish lateness of orders. An order is late if its completion time (computed by $o\_end(\mathcal{M}\_o, \mathsf{t})$) is greater than the due date of the corresponding order (computed by $o\_due(o)$).

Function Defintion 6.2.22

The function o_eval is defined as $o\_eval : \wp(\mathcal{M}_{\nu,\mathsf{t}}) \times \mathcal{O}_{\nu,\mathsf{t}} \rightarrow \mathbb{R}$. For o and $\mathcal{M}\_o, \mathsf{t}$ the function is defined as $o\_eval(\mathcal{M}\_o, \mathsf{t}, o) = \mathsf{x} \in \mathbb{R}$.

If $\mathcal{M}_{\nu,\mathsf{t}}$ is consistent, then the **performance of the network** can be computed, as well. This done by the function ν_eval.

Function Defintion 6.2.23

The function ν_eval is defined as $\nu\_eval : \wp(\mathcal{M}) \times \wp(\mathcal{O}_{\nu,\mathsf{t}}) \rightarrow \mathbb{R}$. For the current, *consistent* set of commitments $\mathcal{M}_{\nu,\mathsf{t}}$ and the current set or orders $\mathcal{O}_{\nu,\mathsf{t}}$ the function is defined as:

$$\nu\_eval(\mathcal{M}_{\nu,\mathsf{t}}, \mathcal{O}_{\nu,\mathsf{t}}) = \sum_{\forall o \in \mathcal{O}_{\nu,\mathsf{t}} \text{ and } \mathcal{M}\_o,\mathsf{t}} o\_eval(\mathcal{M}\_o, \mathsf{t}, o)$$

The commitments of a representative $\mathcal{M}^{\mathcal{R}}{}_{a,\mathsf{t}}$ are **complete** at *pit* t iff

$$\bigwedge_{\forall o \in \mathcal{O}_{a,\mathsf{t}}} m\_comp(\mathcal{M}_{o,\mathsf{t}})$$

The commitments of a representative $\mathcal{M}^{\mathcal{R}}{}_{a,\mathsf{t}}$ are **consistent** at *pit* t iff

$$\bigwedge_{\forall o \in \mathcal{O}_{a,\mathsf{t}}} m\_con(M_{o,\mathsf{t}}, o)$$

6.3. Coordination requirements

After we have modeled the problem space we have to detail the coordination requirements. This is the second step of the ECo process (the context of the ECo process is shown in Figure 6.3.

The requirements of this case study have been derived from the literature of production network like Corsten and Gössinger [CG01] for instance. Similar coordination requirements have been discussed by a number of authors, for instance, Frey et al. [FSWZ03], Dudeck [Dud04], Schut et al. [SKL+04], Jung and Jeong [JJ05], and Schumann et al. [SLT08a].

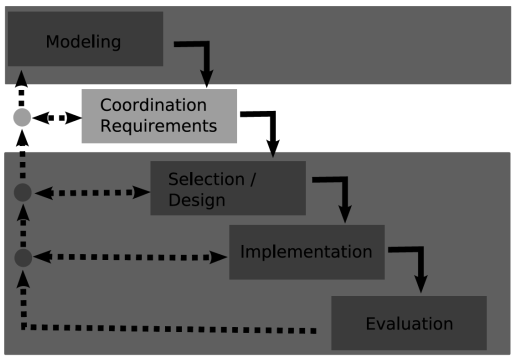


Figure 6.3.: Context of the requirement definition step in the ECo process

In the following the coordination requirements will be formalized that have to be satisfied by a coordination mechanism to be applicable for the coordination of the different planning entities of the production network.

Thereby, a first objective is, of course, that the coordination has to ensure that all local plans and the overall set of commitments of a network, which corresponds to the plan of the network, are consistent as defined above. Additional requirements are listed in the following.

1. Networks have to handle dynamics,
2. Autonomy has to be preserved,
3. Information hiding, and
4. Scheduling systems have to be black boxes.

Networks have to handle dynamics

Dynamics in the context of planning have been discussed in Section 2.2.3. As pointed out before, dynamics are typically modeled as events, that occur and change the environment. Therefore, it is necessary to specify which events can occur and what consequences each type of event has. The dynamics that can occur in this scenario are described by events, as well. In the following we outline the notation of events and what types of events can occur. This defines the degree of dynamics that a coordination mechanism must be able to handle. The events that could happen are:

- an agent could decommit from a given commitment,
- an agent could request a change of its commitment,
- a new job has to be issued,
- a new order arrives,
- an existing order is withdrawn,
- an existing order is changed,

- a new agent enters the network, and
- an agent leaves the network.

Of course, it is arguable if the changes of orders or commitments have to be modeled as separate events, as they could be modeled as a sequence of remove and insert events. But this could lead to unnecessary plan nervousness, as the current schedule or a slight modification of it could incorporate the event, while removing and inserting would affect larger parts of the plan. Therefore, we think that the additional effort in handling more types of events is justifiable. Other possible events that could be modeled, like the changes concerning the available products, or existing variants are not modeled here, as these are tactical decision, that are not addressed here.

Event modeling

Not every event can occur at any time. For instance, the event that an agent leaves a network is not possible if the agent is not a member of this network[3]. Therefore, events are described with a precondition and its effects, similar to the STRIPS notation of actions, outlined in Section 2.2.1. An event is described here with its

- name (shown in $\mathfrak{fraktal}$ characters),
- preconditions (*pre*),
- add effects (+*eff*), and
- delete effects (−*eff*).

Preconditions have to be fulfilled at *pit* $\mathtt{t}-1$ to ensure the event can occur at *pit* $\mathtt{t}$. An event can occur at any *pit* when its precondition is satisfied. An event change the environment. This changes are described by its effects. The effects are divided into *add* and *delete* effects to ease readability. Such a distinction between effects is proposed by Russell and Norvig[RN03, p. 378], as well. To avoid problems similar to the frame problem of planning, we use the so-called STRIPS-assumption, that everything not mentioned in the effects stays the same, see Section 2.2.1 or Russell and Norvig [RN03, p. 378].

Decommitment

By decommiting the presenter or the receiver of a commitment resolves a commitment. The presenter can decide to release the event, if it is reasonable to do so

[3]This could only happen if events, like messages in electronic networks, can overtake each other.

name	$\mathfrak{decommit}(m)$
preconditions	$pre = \left\{ m \in \mathcal{M}_{a1,\mathrm{t}-1} \wedge m \in \mathcal{M^R}_{a2,\mathrm{t}-1} \right\}$
add effects	$+\mathit{eff} = \emptyset$
delete effects	$-\mathit{eff} = \left\{ m \notin \mathcal{M}_{a1,\mathrm{t}} \wedge \right.$ $m \notin \mathcal{M^R}_{a2,\mathrm{t}} \wedge$ $\left. m\_j^i(m) \notin \mathcal{J}^i_{a,\mathrm{t}} \right\}$

Table 6.3.: Summarizing the event decommitment of the commitment m

for him, either if he cannot or will not satisfy the commitment. The commitment receiver can decommit if, for instance, the corresponding instantiated job is not needed anymore. When a decommitment is published, the corresponding instantiated job is removed from the current local schedule. Decommitments have been discussed in Section 3.1.3, too.

Let $a1 = m\_pres(m)$ be the presented of commitment m and $a2 = m\_rec(m)$ be the receiver of commitment m. The event is characterized in Table 6.3.

Change request for a commitment

A change request for a commitment is sent by the commitment presenter to the commitment receiver. Thereby, the agent requests to change aspects of the commitment, for instance, the time interval for the execution. The commitment receiver can decide whether it accepts the change or decline it. This decision is typically made based on the effects that the change would have on the local plan and utility function of the receiver agent. If the request is accepted the corresponding instantiated job and the corresponding assignments have to be updated. Otherwise the commitment remains unchanged. In consequence of a declined change request, the commitment presenter might consider to decommit from the commitment. This, in fact, should be incorporated by the commitment receiver, as well.

Let $a1 = m\_pres(m)$ be the presented of commitment m and $a2 = m\_rec(m)$ be the receiver of commitment m. Let m' be the commitment that is suggested by $a1$ to $a2$ instead of the existing commitment m. Agent $a2$ has to answer to this change request. It can either accept it, exchange the commitment m with m', or it can reject the change request, demanding the fulfilment of m. In the following we assume w.l.o.g. that the instantiated job has been changed to j'^i. Depending on the requested change, it can become necessary to adapt the local

name	$\mathfrak{change\_commitment}(m, m')$
preconditions	$pre = \left\{ m \in \mathcal{M}_{a1,\mathrm{t}-1} \wedge m \in \mathcal{M^R}_{a2,\mathrm{t}-1} \wedge m' \notin \mathcal{M}_{a1,\mathrm{t}} \wedge m' \notin \mathcal{M^R}_{a2,\mathrm{t}} \right\}$
add effects	$+\mathit{eff} = \begin{cases} \emptyset & \text{reject} \\ \left\{ m' \in \mathcal{M}_{a1,\mathrm{t}} \wedge m' \in \mathcal{M^R}_{a2,\mathrm{t}} \wedge j'^i \in \mathcal{J}^i_{a,\mathrm{t}} \right\} & \text{accept} \end{cases}$
delete effects	$-\mathit{eff} := \begin{cases} \emptyset & \text{reject} \\ \left\{ m \notin \mathcal{M}_{a1,\mathrm{t}} \wedge m \notin \mathcal{M^R}_{a2,\mathrm{t}-1} \wedge j^i \notin \mathcal{J}^i_{a,\mathrm{t}} \right\} & \text{accept} \end{cases}$

Table 6.4.: Summarizing the event change request of the commitment m

plan, because the instantiated job j^i has been modified to j'^i. Consequently, there are two alternative effects of this event, depending on the decision of the receiver agent. The event is characterized in Table 6.4.

Instantiated job is tendered

An instantiated job can be tendered, either because an existing commitment has been broken and a new agent for executing the job has to be identified, or commitments for a new order has to be arranged.

If agent a receives a call for proposals for the instantiated job j^i. It evaluates how this additional job would effect its local plan. On the base of this information it might consider to publish a proposal for the instantiated job. All published proposal are collected by the requesting agent and one proposal is selected. The corresponding agent is then informed and is requested to commit to the proposal. If he does so, the tender is successful, otherwise it has failed and is discard.

Assume the instantiated job j^i is tendered. Let ν be the network that the instantiated job origins from. Multiple agents are involved in this interaction, even if communication is always modeled pairwise. Those allocation are typically done in a one-to-many form of negotiation. Let $\mathcal{A}*$ be the set of agents that are

name	$\mathfrak{task\_tender}(j^i)$
preconditions	$pre = \left\{ j^i \in \mathcal{J}^i_{\nu,\mathrm{t}-1} \wedge \notin \mathcal{J}^i_{a,\mathrm{t}-1} \wedge a \in \mathcal{A}^C_{j^i,\mathrm{t}-1} \wedge r \notin \mathcal{R}_{a,\mathrm{t}-1} \right\}$
add effects	$+\mathit{eff} = \begin{cases} \left\{ \forall a*_i \in \mathcal{A}* \, \exists r : r \in \mathcal{R}_{j^i,\mathrm{t}} \wedge r \in \mathcal{R}_{a*_i,\mathrm{t}} \wedge j^i \in \mathcal{J}^i_{a*,\mathrm{t}} \wedge \exists m \in \mathcal{M}_{a*,\mathrm{t}} : m\_j^i(m) = j^i \wedge m \in \mathcal{M}_{\nu,\mathrm{t}} \right\} & \text{tender successful} \\ \emptyset & \text{tender fails} \end{cases}$
delete effects	$-\mathit{eff} = \emptyset$

Table 6.5.: Summarizing the event tender of instantiated job j^i

requested for a proposal for the job j^i. Each agent can answer to an offered job with a proposal. The agent a will publish its proposal r_a if it is advantageous for him, i.e., increasing its utility function. Otherwise, it will not publish a proposal. If the proposal is published, it is evaluated by the requesting agent a_r. Therefore, the agent a_r collects all proposals ($\mathcal{R}_{j^i,\mathrm{t}}$) and selects an appropriate proposal $r*$, using its function $r\_sel(\mathcal{R}_{j^i,\mathrm{t}}, o) = r*$. The corresponding agent $a*$ is informed and gives, if it wants to, its commitment to perform the job. Otherwise the tender is failed. The event of a tender for an instantiated job is formalized in Table 6.5.

New order

Orders are part of the dynamic environment of a network. Dynamics concerning an order are within the scope of the perception of the agent that represents the order. An order o is specified for a specific network ν. Thus, the representing agent a has to be member of the network. If a new order o arrives, this order has to be decomposed into instantiated jobs. Formally, this event can be described as shown in Table 6.6.

In subsequence of this event, the new instantiated jobs have to be scheduled. Thus, a number of $\mathfrak{task\_tender}(j^i)$ events are triggered by this event.

name	$\mathfrak{order\_new}(o)$
preconditions	$pre = \left\{ is\_member(a, \nu, \mathsf{t}-1) = true \right\}$
add effects	$+eff = \left\{ \begin{array}{l} \mathcal{O}_{\nu,\mathsf{t}} = \mathcal{O}_{\nu,\mathsf{t}-1} \cup \{o\} \wedge \mathcal{O}_{a,\mathsf{t}} = \mathcal{O}_{a,\mathsf{t}-1} \cup \{o\} \wedge \\ \mathcal{J}^i_{\nu,\mathsf{t}} = \mathcal{J}^i_{\nu,\mathsf{t}-1} \cup \{\forall j \in v\_j(o\_ref(o)) : j\_inst(j, o, i)\} \end{array} \right\}$
delete effects	$-eff = \emptyset$

Table 6.6.: Summarizing the event new order o

name	$\mathfrak{order\_rem}(o)$
preconditions	$pre = \left\{ o \in \mathcal{O}_{a,\mathsf{t}-1} \right\}$
add effects	$+eff = \emptyset$
delete effects	$-eff = \left\{ \begin{array}{l} o \notin \mathcal{O}_{\nu,\mathsf{t}} \wedge o \notin \mathcal{O}_{a,\mathsf{t}} \wedge \\ \nexists j^i \in \mathcal{J}^i_{\nu,\mathsf{t}} : j^i\_o(j^i) = o \end{array} \right\}$

Table 6.7.: Summarizing the event withdraw of an order o

Order withdrawn

As well as orders can be added they can be removed, too. The representing agent has to react to this event by removing the order and its corresponding instantiated jobs and commitments from the network-wide plan. Let o be the order that has to be removed and be a the representing agent. The formal definition of this event is shown in Table 6.7.

This event triggers a set of ($\mathfrak{decommit}(m)$) events for the effected instantiated jobs, that become unnecessary when the order o is withdrawn.

Order changes

The customer can change its order o to o' before it is fulfilled. As a consequence, the information about orders and instantiated jobs has to be updated. The event is described in Table 6.8.

As a consequence of the change of an order the global (network-wide) schedule is threaten to become inconsistent. The existing plan has to be evaluated and possible inconsistencies have to be removed. It can become necessary that existing commitments have to be changed, or if this is not possible resolved and new

name	$\mathfrak{order\_change}(d, d')$
preconditions	$pre = \left\{ o \in \mathcal{O}_{a,\mathrm{t}-1} \land o' \notin \mathcal{O}_{a,\mathrm{t}-1} \right\}$
add effects	$+\mathit{eff} = \left\{ o' \in \mathcal{O}_{\nu,\mathrm{t}} \land o' \in \mathcal{O}_{a,\mathrm{t}} \land \forall j^i \in \{j\_inst(j, o', i)(v\_j(o\_ref(o)))\} \subset \mathcal{J}^i_{\nu,\mathrm{t}} \right\}$
delete effects	$-\mathit{eff} = \left\{ o \notin \mathcal{O}_{\nu,\mathrm{t}} \land o \notin \mathcal{O}_{a,\mathrm{t}} \land \nexists j^i \in \mathcal{J}^i_{\nu,\mathrm{t}} : j^i\_o(j^i) = o \right\}$

Table 6.8.: Summarizing the event change of an order o

name	$\mathfrak{agent\_enter}(\nu)$
preconditions	$pre = \left\{ \nu\_active(\nu) = true \land is\_member(a, \nu, \mathrm{t}) = false \right\}$
add effects	$+\mathit{eff} = \left\{ is\_member(a, \nu, \mathrm{t}) = true \land \mathcal{B}_{\nu,\mathrm{t}} = \mathcal{B}_{\nu,\mathrm{t}-1} \cup \mathcal{B}_{a,\mathrm{t},\nu} \right\}$
delete effects	$-\mathit{eff} = \emptyset$

Table 6.9.: Summarizing the event an agent a enters the network ν

commitments have to be established. Thus, in the following a set of events of the types: $\mathfrak{decommit}(m)$, $\mathfrak{change\_commitment}(m, m')$ and $\mathfrak{task\_tender}(j^i)$ can occur.

Agent enters network

If an agent a enters network ν, it has to register itself at the corresponding CA. During the registration to the network the agent offers a set of capabilities to the network. Furthermore, it adapts its communication abilities towards the communication protocols used in this network. The event is described in Table 6.9.

name	$\mathfrak{agent\_leave}(a, \nu)$
preconditions	$pre = \Big\{ is\_member(a, n, \mathsf{t}-1) = true \wedge$ $\mathcal{M}_{a,\nu,\mathsf{t}-1} = \emptyset \wedge$ $\nexists m \in \mathcal{M^R}_{a,\mathsf{t}-1} : j^i\_\nu(m\_j^i(m)) = \nu \wedge$ $\nexists j^i \mathcal{J}^i_{a,\mathsf{t}-1} : j^i\_\nu(j^i) = \nu \Big\}$
add effects	$+eff = \emptyset$
delete effects	$-eff = \Big\{ is\_member(a, \nu, \mathsf{t}) = false \wedge$ $\nexists b \in \mathcal{B}_{\nu,\mathsf{t}} : \forall_{a \in \mathcal{A}_{\nu,\mathsf{t}}} b \notin \mathcal{B}_{a,\mathsf{t},\nu} \Big\}$

Table 6.10.: Summarizing the event an agent a leaves the network ν

Agent leaves network

If an agent a leaves the network ν it has to deregister itself from the corresponding CA. This, of course, requires that the agent is already member of this network. Moreover, the agent cannot be a representative of an order of the particular network, and it is not allowed to receive or to given any commitment for instantiated jobs of this network. So prerequisite for finally leaving the network is to disengage the agent a from the network's activities. This could be realized by rising other events like $\mathfrak{decommit}(m)$ or $\mathfrak{order\_rem}(o)$. We model it in this way, because entering and leaving a network are strategic decisions by the planning authority. As we are addressing the operational level here, we do not model events of the tactical or strategical level. As a consequence, the agent has to deregister its capabilities and removes all instantiated jobs and the corresponding assignments and commitments from its local knowledge-base. The event is described in Table 6.10.

Autonomy has to be preserved

In contrast to the rather broad discussion about the term autonomy in general in the research of DAI, see, e.g., Bradshaw et al. [BFJ$^+$04], Luck et al. [LDM03], or Nickles et al. [NRW02], we give a more specialized definition of autonomy here, that has to be preserved. Autonomy of a PAA is characterized by its ability to determine its local schedule, or more precisely the instantiated jobs that are realized with its schedule. An autonomous PAA is in control about which jobs it accepts and how the corresponding proposals and commitments looks like. So in short:

> Autonomy is defined by the fact that the agent controls its jobs and thereby its local schedule.

In consequence autonomy implies that

- the local plan is only determined by the local planning system,
- all modifications of the input data for the planning system are made by the PAA according to its local utility function,
- all decisions the PAA made are documented by the commitments it gives to other agents.

It can be summarized that there is no direct inference from other agents to the local decision making. Of course, other agents can offer compensations, but the final decision, if the agents accept those compensations or other offers are made locally.

As a consequence of this autonomy, the overall system cannot be optimized on the network level. Neither guaranties can be given that each order is processed. For instance, if an agent exclusively offers a certain capability for a job to a network that needs to be executed to fulfill an existing order, this order cannot be executed if the agent is not willing to integrated the corresponding instantiated job into its local schedule. Note that by definition of autonomy there exist no entity except the agent itself that can enforce the agent to integrate that specific instantiated job into its schedule.

A similar notion of autonomy has been used by other researchers investigating the coordination among agents, e.g., von Martial stated:

> "A coordination agent, . . . has to support coordination and cooperation and is not allowed to enforce it." [Mar92, p. 130]

Of course, there might be tactical or strategic considerations at that planning authority, that if the agents lead the network to fail on a specific order that this will have consequences for the reputation of the agent in the network. But these consequences are not subject of this study, because only the operational decision level is taken into consideration here. Those tactical decision influence the negotiation strategy the agent has to follow during the operational coordination.

Information hiding

Information hiding can be seen as a different aspect of autonomy. It is not the control over local decision making and local resources, but having control about the externally available information about the entity itself. This is a special aspect of privacy, that might be characterized as informational self-determination, which origins in the German term “Informationelle Selbstbestimmung”. Information hiding is characterized by Halpern and O’Neil [HO05] in a way that all observers of an agent do not gain any new information while the agent is performing its actions. This privacy concerns have special relevance, as typically the opponent modeling is a typical aspect of the environment model an agent has to reason about, while constructing its future plans. This is typically referred to as *mutual modeling* [Woo09, pp. 170–173].

Halpern and O’Neil [HO05] point out three different aspects that are important for the design or selection of a privacy enforcing mechanism. These aspects are [HO05]:

- What information needs to be hidden?
- Who does it need to be hidden from?
- How well does it need to be hidden?

For instance, within a production network companies can have overlapping competencies. The relation between the participating entities is in between cooperation and competition, see Corsten and Gössinger [CG01, p. 39]. In this situation participating companies have typically concerns to lose knowledge concerning their core competencies or that other companies can benefit from knowing their current abilities. Examples for such sensitive information are production facilities, production times or costs [CG01, p. 39].

Therefore, it can be reasonable to limit the amount of information exchanged between the agents, even if this will lead to an overall plan with lower quality, see D’Amours et al. [DMLS99]. Agents can exchange abstract plans or planning goals if they are not interested in publishing their concrete detailed plans. This approach is, e.g., presented by von Martial [Mar92]. Or they can exchange partial plans, hiding certain aspects or parts of their plans, the planning knowledge, or the complete set of intended goals, which is promoted in the approach of PGP, presented by Durfee and Lesser [DL87], and its successor GPGP, presented by Decker and Lesser [DL92], and Decker [Dec95]. The aspect of information hiding during coordination is also discussed from a game theoretic perspective by Rosenschein and Zlotkin [RZ98b].

Another relevant aspect of information hiding is anonymity, or more precise, the knowledge about which agents participate in which network. In production networks this is typically an aspect that is not intended to make public. There can exist suppliers that are part of competing supply chains [CG01, p. 39]. Thus, even the information in which networks an agent participates in can be a private information. In the following we refer to this aspect as *anonymity.* In contrast to the aspect of *competence protection* that was discussed previously.

Anonymity

As already pointed out, by anonymity we refer to the characteristic that an observing agent a_g cannot infer from its observation that an agent a is member of two networks ν_1 and ν_2. Thereby, we assume that the observer collects the messages send and received from/to agent a. Anonymity is guaranteed, if the agent a_g cannot verify or falsify the result of the following equation.

$$\nexists \mathrm{t} \in \mathbb{N} : a \in \mathcal{A}_{\nu_1,\mathrm{t}} \wedge a \in \mathcal{A}_{\nu_2,\mathrm{t}} \tag{6.3}$$

This definition assumes that the observer a_g uses a boolean reasoning and does not use probabilities for its reasoning. If a_g does so, this definition of anonymity might not be sufficient anymore. Because an observer can have a very high probability assigned to the fact that a is member of ν_1 and ν_2. In this case the definition has to be changed in a way that the observer a_g has no reasons to increase its initial probabilities. So, its confidence that a is member of both networks does not grow, while a_g is observing the interactions within the network.

Research concerning privacy is not in the focus of this study. We do not refine this definition to more elaborated models of privacy preservation. Instead we refer to Halpern and O'Neil who give an interesting overview and introduction in [HO05].

Competence protection

The aim of competence protection is to ensure that for a specific agent a in network ν the other members of this network do not get a precise view of the competences of agent a. Other agents within one network can, of course, see, what instantiated jobs the agent a has committed itself to, if they are commitment receiver of this agent.

In consequence, the information what jobs can be executed by agent a can be collected by an agent receiving a number of commitments from a. This information can be used to approximate the capabilities the agent a offers towards this network. So, this information cannot be kept secret. Of course, by a stricter definition and implementation of anonymity it can be possible to hide this information, too.

More critical information than what jobs an agent can execute is information about the way the agent a executes these jobs. This information contains details about required resources, required processing time, and required process steps. This information is in fact subject of competence protection.

Information about production process can typically be hidden, at least from the perspective of the coordination mechanism. This knowledge is often not modeled explicitly for coordination purposes. Information about available resources should be hidden, as well. For the processing time we have to state a conflicting situation. It is necessary to exchange information about processing times to reach a feasible plan on the network layer. But these published processing times do not have to be equivalent with the processing times required in the internal schedule. Thus, information about the processing time can be noisy, either because an instantiated job could not scheduled within its minimal makespan. Another reason that adds noise in the processing time is the usage of additional buffer to gain more local flexibility, or to obfuscate the real processing time, as part of the negotiation strategy and offered service-levels. Other agents can observe the processing times specified in the commitments. So, if an agent receives a couple of commitments for different instantiations of the same type of job it can approximate the real processing time, with the minimal time offered in the received commitments. So, if $\mathcal{M}'^j$ is the set of commitments for instantiated jobs of the type j, the processing time can be approximated by

$$\mathsf{proc\_time} = \min_{\forall m \in \mathcal{M}'^j} \{m\_due(m) - m\_start(m)\}.$$

Although it can be possible to estimate processing times on the base of additional expert knowledge given to the agents. Thus, the degree of competence protection depends on the strategy the agent a applies in generating the proposals and commitments, derived from its local plan. An offset can be added to the internal duration, to obfuscate the internal processing time. This could lead to less competitive proposals. Therefore, it depends on the local strategy of the agent if, and how much additional time is added. If an offset o is added an observer could, at its best, identify a processing time of $\mathsf{proc\_time} + \mathsf{o}$. Thus, the degree of privacy that can be achieved depends on the one hand on the willingness of the agent to expose information about its processing times and on the other hand on the economic necessity to expose information to generate competitive proposals. The second aspect leads to the fact that current market average processing times are commonly known. Even if an agent b wants to force agent a to reduce its offset, e.g., if agent b compete for the same jobs, this does not change the information balance. Because it is necessary for b to expose or at least publish values more close to its real processing time, to generate more competing proposals, as well. Thus, the situation between competitors is symmetric. Each agent does not know the real processing times, and thus, cannot estimate the risk of exposing its real processing time in competition with other agents with similar competencies.

Scheduling systems have to be black boxes

This requirement follows the design guideline to separate planning and coordination knowledge and mechanisms, which has been suggested by Decker and Lesser [DL93, p. 197]. This guideline leads to a modular architecture containing a scheduling and a coordination mechanism module.

This separation is not widely applied, yet. For instance Dudeck [Dud04] and Frey et al. [FSWZ03] try to take advantage of the integration of local planning systems and coordination mechanism, for the overall coordination between the entities. A pretended disadvantage of the modularization is that no internal information of planning system, like available free capacities in the future, can be used to compute alternative solutions for requests from other agents. This can lead to solutions that are less efficient than solutions generated by mechanisms without modularization. But as we have stated that autonomy preservation and information hiding are important coordination requirements, too. These disadvantage is not significant. As the usage of such private information in the coordination process would violate these requirements.

A possible disadvantage is that all computations concerning the effects of changes to the local plan have to be evaluated by generating a new plan using the planning systems. Thus, the planning system could run multiple times generating plans that are not used in production. Depending on the needed computation time this can become prohibitive.

In contrast the modularization has a number of advantages that enable the usage of automated coordination mechanisms, at all.

First of all, the coordination mechanisms does not have to imply restrictions on the method that is used to generate local plans. This happened, for instance, in the mechanism proposed by Dudeck [Dud04], where all local planning systems are assumed to be mathematical optimization problems. By using such an assumption the pre-existence of planning systems in companies is neglected. As already mentioned, we are addressing here coordination mechanisms for the operative coordination among planning entities. The decision which type of planning systems, a customized 3rd party software systems or a custom software is used, are tactical decision that should not be effected by systems that offer additional capabilities for the operational level. Moreover, a modular approach facilitates to exchange the subsystems independently. Thus, a coordination mechanism can be exchanged, while the local planning system stay in use, and vice versa. According to Gamma et al. [GHJV94, pp. 24,25], this is a consequence of loose coupling of systems in software engineering. A planning authority can also use one planning system with different coordination mechanisms, which could be in place for the coordination in different networks.

As already indicated, planning systems are often pre-existing at the time it becomes reasonable to integrate a coordination mechanism. Therefore, a modular approach facilitates to use existing planning systems. These planning systems are are solely designed to generate goods local plans for the situation in the particular problem domain. An existing planning system is a single source of information for the current control of the production of goods or services, that has been established to achieve effective value creation, see Kurbel [Kur05, pp. 4–8]. The design of such an infrastructure is a strategic and tactical goal of the IT-management, see Hansen and Neumann [HN09, p. 240]. Using a modular approaches enables to preserve this infrastructure and to not effect the decisions of the tactical and strategic decision level.

In consequence the coordination mechanisms should only have the ability to define input data for a planning system, start the planning system, and read and interpret the resulting plans. Interactions between scheduling and coordination modules have been detailed, for instance, in Decker [Dec95, Chap. 5.7] for the GPGP coordination mechanism.

6.4. Selection of appropriate coordination mechanisms

After the coordination requirements have been defined the third step of the ECo process is the selection step.

The overall context of this step in the ECo process is given in Figure 6.4. In Section 4.2.3 (on page 130) the following techniques have been identified as applicable for this scenario, according to the characteristics of coordination problems identified in Section 4.1 (on page 117). These mechanisms are listed below:

- mediator-based,
- auctions, and
- negotiation.

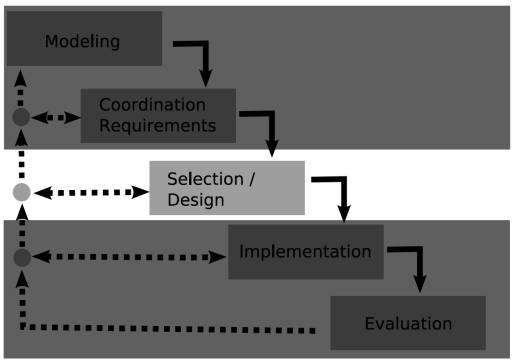


Figure 6.4.: Context of the selection step in the ECo process

In the following we discuss these techniques in the context of the coordination problem of this case study and then identify the most suitable approaches that can be used in this case study. The field of coordination in production networks and supply chains has been issued by a number of researchers, therefore, we cannot claim a complete overview, even if we present a wide survey.

Mediator-based approaches

Mediator-based approaches have been discussed in Section 3.1.2. A main idea of these approaches is that inconsistencies or conflicts between two or more local plans can be resolved by a centralized decision maker, called mediator. Therefore, the concept is similar to the bottom-up planning approaches, discussed in the context of coordination approaches from the field of business administration, see Section 3.1.1.

To implement such an approach, it is necessary that the mediator has enough knowledge about the local entities to either generate non-conflicting plans that are feasible at the local level, or at least can provide restrictions to the local plans that enforce global and local feasibility. Additionally, the mediator does not only requires the knowledge to generate a solution, but also requires the authority to enforce this mediated solution. That is, the local planning entities have to apply the plans or constraints given by the mediator.

An approach where a technique similar to mediators is used has been presented by Naso and Turchiano [NT04]. In their study a production process in analyzed. Decision making is distributed among parts and workstations. Parts are represented by part agents that are responsible for routing the part through the production process. Workstation that can perform actions in the production process are represented by workstation agents. They can either accept or reject requests from parts, or determine the sequence of parts that have been accepted for production. Therefore, the modeling with part and workstation agents is similar to work presented by Schumann and Sauer [SS07]. In their work Naso and Turchiano allow that workstation agent can reason about conflicts and performance lacks due to local decision making by the part agents that leads to inefficient routings. The workstation agents can then build new routings for parts and communicate those routings to the part agents, that have to follow their new route.

Auctions

Auctions, as a technique for coordination, have been discussed in Section 3.1.3. Within a supply chain the coordination problem is similar to solving a scheduling problem by assigning instantiated jobs to agents that executes these jobs. As it is necessary to execute all instantiated jobs that belong to an order, it can be stated that the assignments are complementarity and a multi-item auction has to be performed. According to Schmidt [Sch99] combinatorial auctions, presented in Section 3.1.3, are an adequate mean. Additonal complexity is added as a correct sequence among the scheduled jobs has to be guaranteed. Sequencing considerations are not part of standard techniques for combinatorial auctions. In fact the usage of combinatorial auctions for scheduling problems is subject of

r1	**r1**	**r1**	**nop**	**nop**	**r2**	**r2**	**r3**

Figure 6.5.: Example of a bid structure with 7 time slots and 3 resource (r1,r2,r3)

research itself, e.g., Wellman et al. [WWWMM01] or Elendner [Ele04].

A technique for solving distributed scheduling problems that has been advocated for in the literature [WWWMM01, Ele04] is based on the auctioneering of time slot of the resources. In the following we describe the specifics for the model presented above. Each bundle contain all time slots for the time horizon of the scheduling. This horizon has to be bounded, which is in fact not a very restricting assumption, as each schedule is bounded by a time horizon. Lets assume the planning horizon contains t time units. Then each bundle would consists of t elements. Each slot in a bundle can have an identifier of a resource or a *no-op*. Whereby a no-op indicate that no resource is occupied by that bundle at the specific time slot. If there are r possible resources, then for each slot in a bundle $\mathsf{r}+1$ values are possible. With this notation it is indicated that at a specific time slot (the corresponding slot within the bundle) this bundles offers the usage of the resource, that is specified in the corresponding slot. According to the model we have described above, a resource is characterized by the execution of a specific job at a specific planning entity. Consequently, the execution of a job at agent a and b would have to be modeled as two different resources. An example of 7 time slots and 3 resources a bid is exemplified in Figure 6.5

The orders, or in our model, their representative have to bid for each bundle. Therefore, they have to evaluate for each bundle if the order could be realized, and if so, compute how efficient the order is processed by the corresponding assignment. This could be realized by computing the price for a bundle as the difference between the maximal end of the jobs required for an order o and its due date. If it is not possible to execute an order with a given bundle then the bid should have a maximal negative value.

If for each order bids have been collected the winner determination has to be done. The winner determination problem is a selection among the bids whereby for each order exactly one bid has to be assigned, while maximizing the value of the selected bids. The winner determination problem becomes simpler in this type of auction, because all feasibility checks have been performed during the bidding process and are not part of the winner determination problem.

In this auction process it is assumed that all resources are available at each time slot. Which can only taken for granted, if the resources are not used for other jobs than those of the particular network. Or, in consequence, for the model presented here, the planning entity cannot coordinate their activities within more than one network at time. But the main disadvantage of this procedure is the huge amount of possible bids. With r resources for t time slots there exists $(\mathsf{r}+1)^{\mathsf{t}}$ different bundles. In consequence, this process can only be done for very cause grained time intervals and even then becomes very fast prohibitive complex. Even though most bundles do not lead to a feasible plan for an order they all have to be evaluated during the bidding process (by each bidder) and during the winner determination.

Another way to model the coordination problem of the production network using auctions is described in the following. Bundles are members of the powerset of the instantiated jobs that have to be performed. For each bundle the bid is not a simple scalar anymore but a set of price vector. A price vector is a vector that contains for every time slot a price that has to be paid. An example of the usage of price vectors is presented by Grolik et al. [GSW$^+$01]. A bid has a set of vector. One price vector for each element of the bundle. Bids are given by the planning entities, encoding the fact that it can offer the service at a specified time for a specified price. Thus, the PAA can distinct between times when its resources are not available, or it is advantageous to behave opportunistic. This enables the coordination within different networks, sequentially. Assuming we want to generate a schedule for the next t time slots with n instantiated jobs that have to be performed to execute all orders. The resulting number of bundles per bidder is $2^{\mathsf{n}} - 1$. And consequently, $\mathsf{k} * (2^{\mathsf{n}} - 1)$ bids have to be evaluated by the auctioneer. It becomes clear that except for very small number of time slots the number of bundles will be smaller than in the second model. But this leads to a more complicated winner determination. As the sequencing and durations of the different jobs has to be regarded during the winner determination problem. Moreover, it has to be taken care of the fact that, if possible, the dues for the orders have to be satisfied.

So far, strategic bidding has not been addressed in this auctioning protocols, at all. Wellman et al. [WWWMM01, p. 299] have investigated that the generalized Vickrey auctions could be modified for distributed scheduling, as well. But this will result in the fact that the winner determination problem, which is a hard problem, has to be solved multiple times. Thus, this becomes typically prohibitive complex.

In summary for the multi-item auction scenarios the resulting complexity, for a scenario like the one presented here, is enormous. The problems used in the literature, e.g., by Wellman et al. and Elendner [WWWMM01, Ele04] typically have a simpler structure. While auctions are efficient for single-item allocation problems, this cannot be stated for multi-item allocation or scheduling problems. This observation is supported by Wellman et al. [WWWMM01, p. 300], as well. The repetitive usage of single-item auctions for solving a multi-item problem is neither an option as no guarantees can be given concerning the solution quality, see [WWWMM01], and therefore auctions are not superior to other approaches.

Negotiation

Negotiations in multiagent systems are a widely covered area in DAI research and have been outlined in Section 3.1.2. Consequently, various negotiation-based mechanisms for coordination for scheduling problems and for the coordination in production networks, like supply chains, have been proposed.

Most of these approaches assume collaborative, or at least truth telling agents. An exception is the work presented by Sandholm and Lesser [SL95]. The main idea is to coordinate plans by information exchange that is regulated by conversation protocols. This, of course, require that all agents are aware of these protocols and that they share a common ontology to interpret the messages that are exchanged, correctly. The mechanisms proposed for conflict resolution vary from hierarchies to bargaining.

A simple forms of a negotiation protocol is the contract net protocol (see Section 3.1.2). One of the first investigations concerning the usage of the contract net protocol for the coordination in scheduling problems has been presented by Parunak [Par87]. Within the YAMS system local scheduling entities negotiate to generate a manufacturing plan within a volatile environment.

Sandholm and Lesser [SL95] argued, that the contract net protocol has been developed for cooperate problem solving problems, and that it is not well suited for production networks comprising of autonomous entities, like for instance virtual enterprises or supply chains. They argue that for representing autonomous entities self-interested agents are more suitable. Therefore, they have extended the contract net protocol. In this context they have also introduced the concepts of leveled commitments and decommitment penalties (see Section 3.1.3).

With the MAGNET system presented by Collins et al.[CTMG98], the interaction between the negotiating partners is realized by a market framework that enables indirect negotiation. This could be used, e.g., to foster information hiding aspects and ease the development of agents, as the MAGNET system offers some services required for the negotiation.

A specific approach for negotiation-based coordination in the context of supply chains has been presented by Barbuceanu et al. [BTF97]. Their approach base on the specialized coordination language COOL, an extension of the KQML language, see Section 2.3.4. An introduction into the COOL language is given by Barbuceanu and Fox [BF95]. The planning process is organized as a sequence of communication processes that pass a partial plan to an upstream entity. Each type of conversation is regulated by a specific rule set. Thus, the plans of each local entity is communicated upstream the supply chain. In consequence, a distributed planning is realized for a joint plan as introduced in Section 3.1.2.

Wagner et al. [WGP03] present an approach for the coordination within a supply chain based on TÆMS (see Section 3.1.2). Therefore, the production alternatives are build as TÆMS structures and existing interdependencies have been added. The conventional GPGP coordination mechanism is extended by the integration of commitments, decommitments and decommitment penalties as introduced in Section 3.1.3. These penalties are computed using internal costs of the planning authorities. The focus in their paper is on tactical planning, decisions concerning the product mix are coordinated. Consequently, their approach does not have to deal with the integration of pre-existing planning systems as they are operating on more aggregated data.

The SPP "RealAgentS" has contributed significantly to the research concerning the coordination of distributed planning systems. In particular in the Agent.Enterprise, presented by Woelk et al. [WRZN06]. In this effort different planning systems have been designed that have to coordinate their local plans. These coordination mechanism has been described by Grolik et al. [GSW$^+$01], and Frey et al [FSWZ03], for instance.

Moreover, in a another project of this research effort the coordination within medical units in a hospital has been investigated by Paulussen et al. [PJDH03], which based on negotiation schemes, as well.

Qualitative evaluation

In this qualitative evaluation we discuss the different approaches sketched above. In particular we discuss their abilities to satisfy the coordination requirements. Due to the large number of approaches presented here, this is done in a less formal way. We point out if an approaches violates a coordination requirements and if so, why.

Mediator-based approaches

The fact that the mediator requires knowledge about local abilities to generate solutions or constraints that lead to feasible plans on the local level is in conflict with the coordination requirement concerning information hiding. Aspects like processing times and current workloads have to be accessible for the mediator, at least in an abstract way. This includes restrictions to the local plan imposed by commitments given for contribution to other networks or for network unrelated jobs. This is a serious restriction concerning the information hiding requirement. Additionally, the mediator has to have the authority to implement the mediated plans. This is in conflict with the coordination requirement concerning the local autonomy of the planning entities. This requirement stated that the planning entity has full control of the jobs it accepts.

In summary, we see that the mediator-based coordination conflicts with the coordination requirements concerning information hiding and preserving local autonomy. Therefore, we do not see that approaches based on this concept are suitable for the coordination of production networks, addressed here.

Auctions

As already discussed in Section 3.1.3, the advantage of a combinatorial auction is its ability to provide high-quality or even optimal solutions. But the computation of bids for each bundle can be very time consuming and the winner determination is a hard problem, as well.

Concerning the coordination requirements auctions do fulfill most aspects. Depending on the implementation of the auction the coordination within multiple networks can become a problem. In a situation in which the time slots of the resources are auctioned the local autonomy is lost, as the assignment is done by a centralized winner determination entity and not by the planning entities anymore. Other approaches do not necessarily violate this requirements, for instance, when planning entities offer price vectors for services.

Concerning the aspect of information hiding a differentiated analysis has to be done. If the auction is done based on time slots for resources, the bidding agents, the representatives of order, requires knowledge about the processing times for each job to compute if it is feasible. Using a price-vector based approach in the design of bids knowledge about feasibility is required to solve the winner determination problem. Consequently, both approaches require this information. In the one case it has to be distributed among the representative agents, in the other it has *only* made accessible to one centralized entity. In both cases the requirement of information hiding cannot be satisfied completely.

The separation of coordination and planning systems cannot be implemented in an auction based coordination scheme. The local plan can simply be derived from the assignments made in the auctioning process. A planning system might be used for the price calculation for the offered price vectors, but the final assignment is made during the auctioning process.

Moreover, events that change the planning environment have to be infrequent. Infrequent in relation to the time required to compute a solution for the combinatorial auction. If an event changes the planing situation of one bidder it has to update its bid. All other bids are assumed to remain valid. After an update a new winner determination has to be performed. It becomes more critical when an event changes the number of possible bundles, in this situation the entire auctioning process has to be repeated. The aspect that handling dynamic requires a lot of computational resources and can lead to a complete re-computation of the entire overall plan is critical for the satisfaction of the coordination requirements to handle dynamics.

As pointed out, the number of bundles growth enormous. For instance, for the production scenario used in this case studies each order specifies one type of product, and four operations have to be scheduled. Each operation is an element of a bundle, as it can be allocated at one of the two companies, at least for product 2. So, if n is the number of goods, here operations, the number of bundles is $2^{\mathsf{n}} - 1$.

Up to now it was of no importance how the local planning process looks like. Now, as we want to point out the complexity of generating one bid, this becomes an issue. Therefore, we have to detail our case study. At this point in time we use the simplifying assumption that both planning system have to solve very similar scheduling problems. We relax this assumption later, and allow different companies to have different local scheduling problems. Another restriction we made for the first part of our investigation is that the companies do not have overlapping competences, i.e., they are not able to offer operations that are provided by another company. Only for product A in our case study the required operations are distributed without overlap. Thus, we limit orders only to the first product type. This limiting assumption will also be relaxed in the following. We assume that each operation can be achieved by scheduling two actions sequentially. Each action requires processing time on a local resource. To keep the local scheduling problems simple, we assume that each action can be performed only on one resource and execution time is fixed. We assume that there exist five different resources at each company. As we have outlined above, we currently limit the case study to orders requiring product 1. Thus, we detail only the production planning problems for this product. In the following we present more complex local planning problems. The details of these simplified local planning problems are provided in Table 6.11. We use the naming convention that resources of company A are enumerated from $1 \ldots 5$ and for company B we enumerate them from $6 \ldots 10$.

Both planning approaches can be implemented, e.g., by a constraint solver, which has been done in the work the ABACO approach was originally presented in, see Schumann [Sch04], and Schumann and Sauer [SS05].

If the computation for the generation of one bid takes three second, which is according to our tests a reasonable amount of time, the total computation time for all bids for one bidder can be approximated by multiplying the number of bundles with the time needed to compute one bid, as it is necessary to compute a bid for each bundle. The resulting needed computation time for bids for different

Company A			
operation	activity	resource	duration
111	1	1	3
	2	3	2
114	1	3	2
	2	5	3
Company B			
112	1	6	5
	2	7	5
113	1	6	5
	2	8	5

Table 6.11.: Details of local production details for SC operations

orders	number of bundles	approx. computation time for bids
1	$2^4 - 1 = 15$	45 sec.
2	$2^8 - 1 = 255$	12,75 min
3	$2^{12} - 1 = 4095$	3,4 h
4	$2^{16} - 1 = 65535$	>2,2 days
5	$2^{20} - 1 = 1.048.575$	36,4 days
...	...	...
10	$2^{40} - 1 = 1.099.511.627.775$	>104.595 years

Table 6.12.: Growth of complexity for computing bids for a CA

problem sizes is shown in Table 6.12.

Consequently, the combinatorial auctions is only reasonable for a small number of orders. Which is unreasonable, because for such small instances, plans can even generated manually. So, we can summarize that for auctions only a number of coordination requirements can be partially satisfied and the complexity of this mechanism is a serious problem.

Negotiations

At a first glance negotiations seem to fit very well to the needs of coordination of planning systems, which may be a reason for the large fraction of negotiation-based approaches presented in this field of research. But a detailed analysis has to be performed. As indicated in earlier work, published in Schumann et al. [SLT08a], most of these approaches fail to match all coordination requirements.

The concept of negotiation protocols is suited for coordination purposes, as they are not in conflict with the coordination requirements in general. But a detailed analysis is necessary, as aspects like which data is used and published, depends on the concrete implementation.

The coordination approach presented by Barbucenau et al. [BTF97] is based on the idea that local plans are distributed to come to an efficient global plan for the entire supply chain. Thus, aspects like information hiding are not regarded.

Even if the mechanism of Wagner et al. [WGP03] was designed for a tactical problem in the context of supply chains, it allows to clarify one aspect. The design of penalties for decommitment can be particular critical in the sense that for the computation of those penalties typically information is required that is private for each planning entity. Thus, it has to be critically evaluated if the aspect of information hiding is respected in the computation and in the publication of decommitment penalties, as well.

The DISPOWEB approach presented by Frey et al. and Grolik et al. [FSWZ03, GSW$^+$01] applies negotiation among agents to coordinate local planning systems within a supply chain. To foster the performance of the approach information from local scheduling systems is used. All local planning systems are agent-based planning and scheduling systems that have been developed in the "RealAgentS" research effort, as well. A special type of agent is inserted into each planning systems and collects information of the local schedule. This information is used within the negotiation protocol to compute the pricing. Consequently, the coordination requirement that planning and coordination functionalities should be separated, i.e., scheduling systems have to be black boxes, is not regard.

If existing mechanism fail requirements concerning the information hiding or the modularity requirement that scheduling systems have to be black boxes, this is often motivated by the fact that those approaches strive to achieve higher quality solutions from the overall perspective. A consequence of this has been the development of the ABACO approach. It is a negotiation-based coordination approach that achieves these requirements, see Schumann and Sauer [SS05]. Within the ABACO approach a network consists of a number of planning agents, each representing a planning entity, and one coordination agent, as outlined above. For an initial assignment of jobs to agents ABACO uses a negotiation-based approach similar to the contract net protocol. To handle dynamics a number of additional conversation protocols have been defined. In total the ABACO approach can handle the following events, see Schumann [Sch04, p. 114]:

- Agents can change (add and remove) the capabilities they offer to the network.

- Agents capabilities can be changed on the local level.
- Agents can enter and leave networks.
- Agents can change their conversation strategies at runtime.
- Planning entities can be in multiple networks at the same time. Although they can have non-network related orders, that can be added, removed or changed.
- Network orders can be added, withdrawn or changed.
- Variants can be added, removed or changed.
- Products can be added or removed.

Thus, the ABACO approach can handle all kinds of dynamics that have been pointed out above. Moreover, the ABACO approach offers techniques for reacting to more event that are either subject to local decision making or tactical decision making, both not discussed here.

Each ABACO agent has its own utility function that determines for which jobs a proposal is generated, or if a change commitment request is granted or not. The planning entity has to specify guidelines concerning what services are offered to which network, and to what degree concessions are made to whom. There exists no entity that can enforce an agent to perform a job. Thus, the local autonomy, as outlined in the coordination requirements, is satisfied.

The information that is published to each network comprises what kinds of jobs each agent offers to the network. This information is not broadcasted, but only send to the coordination agent of the network. Thus, if this entity can be realized as a thrusted third party entity, this information can be kept confidential. But each agent can be engaged in negotiation with an agent performing a direct preceding or succeeding job for a given order. Thus, each agent can collect information about capabilities. In an offer or a commitment an agent specifies a time window for the earliest start and the latest due of a job. Within this interval the job will be processed. Thus, the execution time can be estimated by the length of this interval. As discussed above, it is possible for the planning entity to specify a network specific offset that is added to the real execution time of the job, that is added to the compute the length of the published interval. Advantages and disadvantages of these information publication strategy have been discussed above.

The access to the local planning systems is realized using files for data exchanged. The PAA uses a wrapper that generates an input file for the planning system. The agent starts the planning system using a script. The planning system generates an output file that can be read and interpreted by the agent using a wrapper again. Thus, the planning mechanism is encapsulated within the planning system, exclusive. The agent can only access the planning functionalities using wrappers. Consequently, the modular approach is implemented within the ABACO approach. Therefore, we can state that the ABACO approach satisfy all coordination requirements.

6.5. Implementation of coordination mechanisms

In this section we discuss the fourth step of the ECo process. The context of this process step in the ECO process is given in Figure 6.6.

In our previous discussion we have pointed out that negotiation mechanisms, especially the ABACO approach enables an efficient coordination, respecting all coordination requirements. Consequently, we reuse the existing implementation of the ABACO approach to implement the coordination process among existing planning systems. Thus, the CoPS process and framework are not applied in this case study.

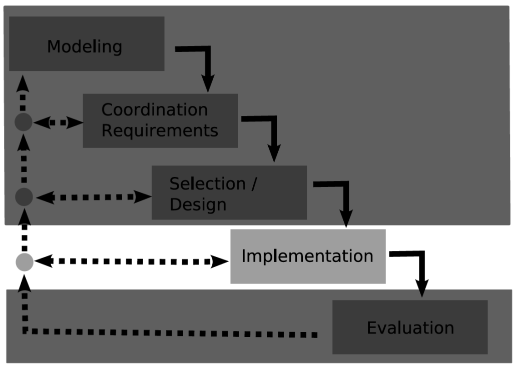


Figure 6.6.: Context of the implementation step in the ECo process

For comparing the results of the ABACO approach we use two different mechanisms, both rely on centralized problem solution. The first one computes the optimal solution using a linear program. Linear programming has been outlined in Section 2.2.1. Linear programming enables us to find the optimal solution, they required computation time limits this approach only to smaller problem instances. Scalability of the coordination approach, in terms of problem size and required computational effort, are discussed below. But for finding optimal solutions the problem size is a limiting factor. For that reason we present another approach for evaluation, as well. The second approach is an order-based heuristic.

The ABACO approach

As the ABACO has been already implemented we are using the existing implementation. Thus, the CoPS process and framework are not used here. But the ABACO approach is a forerunner of the CoPS approach. Work on the ABACO approach has influenced the CoPS process and framework. Similarities especially to the CoPS process are discussed below.

The overall architecture of a network, which uses the ABACO approach for coordination contains a set of PAAs and one coordination server. A PAA represents a planning authority and adapts an existing planning system. A coordination server is an entity, possibly an agent, that offers services for the coordination within the network. The overall architecture is depicted in Figure 6.7.

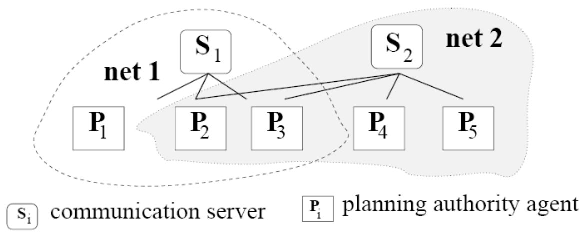


Figure 6.7.: ABACO representation of a production network, [SS05]

Each PAA comprises out of five components. These components are sketched in Figure 6.8. In this graph of the architecture it becomes also obvious that the ABACO approach is a forerunner of the CoPS framework, as the design of a PAA in the CoPS framework (Figure 4.16 on page 159) has similar components. The communication facilities of an agents is externalized in a separate module. Data for the initialization and results, i.e., the local plan, are stored in a database and for reasoning a world model is provided.

In the localization modules aspects like the localization of conversation protocols and adaptation of planning systems, both important issues of the CoPS process, are provided. In the ABACO approach conversation protocols have been

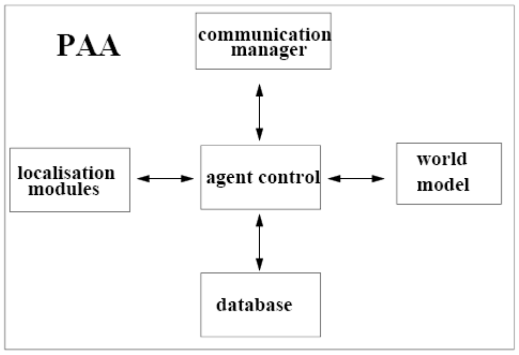


Figure 6.8.: Architecture of a ABACO PAA agent, [SS05]

implemented. The specification of a conversation policy is done by setting parameters to localize the agent to its planning entity. This is done by specifying what jobs/capabilities are offered to which networks, what concessions can be made to whom, and which offset has to be added for proposals that are computed for a given network. These parameters are used in the conversation behaviors that are implemented in the ABACO agents. As outlined above, the adaptation of planning systems is done based on file exchange and additional wrappers. Thereby, the ABACO agents are only capable to adapt local scheduling systems. Additional planning knowledge can be incorporated to facilitate the search for local plan improvements.

The ABACO approach offers three different coordination schemes for different planning situations. It differentiate between:

- predictive scheduling,
- reactive scheduling, and
- iterative plan improvement.

The task of the predictive scheduling is the initial generation of a schedule. Thus, the input of the predictive scheduling is a set of orders and its output is a schedule that represents a feasible plan to fulfill the orders. Within the ABACO approach the predictive planning is implemented in a simple way. Orders are scheduled in a linear way. For each order, each variant is evaluated. The best variant is chosen and commitments are collected from each PAA to whom an activity is assigned to. Note that we have defined only one variant per product here, to keep the example simple, even though the ABACO approach could handle more complex situations.

In contrast, for reactive scheduling it is assumes that there already exists a schedule that has to be adapted to a change in the environment. Such changes are described as events. Thus, reactive scheduling is one particular technique for planning in dynamic environments, see Section 2.2.3. In the ABACO approach it is assumed that reactive scheduling is the common/frequent case, while the initial plan formation is a rather infrequent task. This assumption has been influenced by Henseler [Hen98, p. 31]. Henseler even stated that predictive scheduling is a special instance of the reactive scheduling problem. The ABACO strategy for reactive scheduling is to look first for alternative resources that could perform an activity that is effected by an event. If the event cannot be handled by picking another resource all orders effected by the event are removed from the current schedule and then are re-scheduled on the base of the updated world state.

The idea of iterative plan improvement is to dampen the effect that the plan quality decreases with the number of reactive planning steps, which has been stated by Henseler [Hen98, p. 42]. Therefore, the local system tries to improve

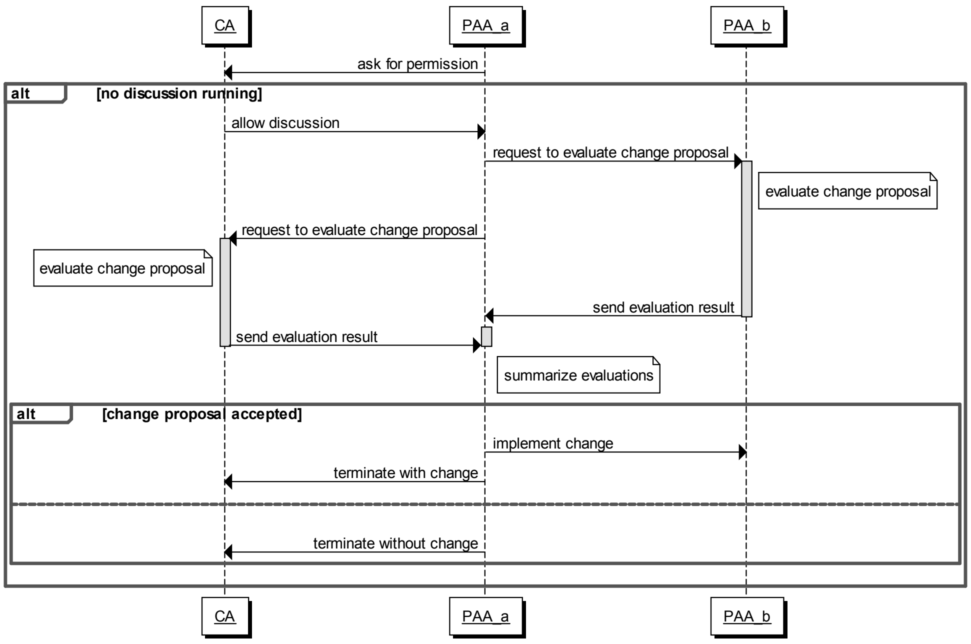


Figure 6.9.: Conversation protocol of an improvement discussion, according to [SS05]

its plan automatically. Thus, a part of the localization module is a sub-system for finding for local plan improvements. If an agent `PAA_a` has found an improvement by changing a commitment it has given to an agents performing an activity preceding or succeeding the activity of `PAA_a`. In the ABACO context a conversation for evaluating if a particular improvement can be implemented is called *plan improvement discussion*, or for short discussion. A conversation flow of one discussion is sketched in the sequence chart presented in Figure 6.9. At first, the agent `PAA_a` has to request the corresponding CA of the network for an allowance to start an improvement discussion. This is necessary to avoid multiple discussions at the same time, which could lead to a simultaneous discussion about all commitments, which is not desirable for complexity reasons. If no other discussion is ongoing the `CA` can grant the request, otherwise the conversation ends. Assume the `CA` has given the allowance for a discussion. The agent `PAA_a` then sent a change proposal to the agents that are responsible for the preceding and succeeding activities. If either does not exist, it sends its request to the CA, who is responsible for ensuring start and due date of the overall corresponding order. Both entities that have received the request have to evaluate the consequences on their local plan of the change that has been proposed. Each agent has three options to answer to a request:

- refuse,
- accept and contribute, or
- conditionally accept.

The agent `PAA_a` collects all answers. If one agent has refused the request the discussion has to be terminated without the change is implemented. If a requested agents favors the change, because it enables an improvement of its local plan, as well, it can accept the change proposal. Moreover, it also can offer to contribute to the implementation of the change with a fixed amount of utility. In contrast, if an agent accepts conditionally it request some utility compensation for the implementation of the change. The agent `PAA_a` has to evaluated if the requested utility can be *paid* by offered and gained utility. If the requested utility for compensation is larger than offered and gained utility, the discussion ends without a change, as it is not possible to leverage the overall performance to a higher Pareto level. If gained and offered utility of a change are greater than the requested compensation, the change can be implemented. The agent `PAA_a` informs the participating agents and the CA that the change is implemented. All participating agents then update their local plan. The CA updates the information concerning the global plan. This message also informs the CA that the discussion has terminated successfully.

For more details concerning the ABACO approach we refer to Schumann [Sch04], and Schumann and Sauer [SS05].

Optimal solutions by linear programming

Within linear programming a maximization or minimization problem is solved, respecting a number of linear constraints. Thus, it is critical to define an objective function. We use a modified version of the total tardiness of all orders presented by Pinedo [Pin05, p. 29]. The tardiness of an order o depends on its completion time c_o, i.e., the time when the order leaves the system, and the due date of the order d_o. Conventionally the tardiness of order o is computed as follows:

$$\mathsf{t}_o = \max\{\mathsf{c}_o - \mathsf{d}_o, 0\}$$

. The total tardiness of all orders are consequently the sum of the tardiness of each order:

$$\sum_{j=1}^{n} \mathsf{t}_j$$

In contrast to the aforementioned definition of tardiness we use a slightly different notation, that enable us to be more distinctive between different solutions/plans.

We use the following notion for the tardiness of an order o:

$$\mathsf{t'}_o = \mathsf{d}_o - \mathsf{c}_o$$

$\mathsf{T'}_o$ is positive, if the job is finished before its due date and becomes negative otherwise. If we want to evaluate all orders, so we sum our modified tardiness.

$$\sum_{j=1}^{n} \mathsf{t'}_j$$

The higher the value of the modified tardiness is, the more compact is the schedule.

The constraints that have to be regarded ensure that all activities of an order are performed sequentially and that all resource capacities are regarded, i.e., each resource is used by maximal one activity per time unit.

For the formulation of the linear program(LP) we use the Mathematical Programming Language (MPL) defined for the *MPL Modeling System* a tool from Maximal Software [4]. A simple notation of a LP for a scheduling problem is shown in Listing 6.1 and discussed in the following.

In lines 2–5 three different indices are defined that are used in the model. For each order one index is required (lines 3 and 4) plus an additional index enumerating the number of orders. In the following lines (6–13) input data is provided. In lines 6 and 7 the duration per activity and order is given. In line 9 the due dates for the orders are specified. In lines 10–13 a binary 8×8 matrix is shown that indicates if different activities require the same resource (indicated by 1). In lines 15–18 the decision variables of the model are defined. In lines 15 and 16 a vector is defined, representing the vector of start times for the activities of an order. In lines 17 and 18 two 8×8 matrices are defined. They are used to ensure the capacity constraints, encoding which activity precedes another. For each pair of orders two matrices are required. In lines 20 and 21 the objective function, discussed above, is defined that has to be maximized. In lines 23–27 the resource capacity constraints are defined. In line 23 the precedence matrices are activated or deactivated. They are relevant if both activities (referred to by their index) use the same resource, which is given in the data block. If this is true, exactly one of the cells in the precedence matrices has to be 1. Lines 24 and 25 encodes that if the activity from two orders use the same resource and the activity of the first order is executed before the second order, the start time of the second activity has to be greater or equal than the end time of the first activity. The corresponding cell value of the decision variable `pre10` is used to encode the condition that the activity of order 2 is not before the activity of order1. Note that the constant 712 is a huge value, that is always greater than the formulae on the other side of the

[4] `http://www.maximalsoftware.com/`, Accessed: 01/06/2010

Listing 6.1 Linear program formalizing the scheduling problem for two orders

```
TITLE Schedulings11
INDEX
pr0 = 1..8;
pr1 = 1..8;
order = 1..2;
DATA
dur0[pr0] := (11,7,19,19,19,19,7,11);
dur1[pr1] := (11,7,19,19,19,19,7,11);
dueDates[order]:=(88,178);
uSR01[pr0,pr1]:=((1,0,0,0,0,0,0,0), (0,1,0,0,0,0,1,0),
                 (0,0,1,0,1,0,0,0), (0,0,0,1,0,0,0,0),
                 (0,0,1,0,1,0,0,0), (0,0,0,0,0,1,0,0),
                 (0,1,0,0,0,0,1,0), (0,0,0,0,0,0,0,1));
DECISION VARIABLES
start0[pr0]
start1[pr1]
pre01[pr0,pr1]
pre10[pr0,pr1]
MODEL
 MAX due =(dueDates[order:=0]-start0[pr0:=8]+du0[pr0:=8])+
        (dueDates[order:=1]-start1[pr1:=8]+dur1[pr1:=8]);
SUBJECT TO
h01[pr0,pr1]:pre01[pr0,pr1] + pre10[pr0,pr1] = uSR01[pr0,pr1];
res01[pr0,pr1]:uSR01[pr0,pr1]*(start0[pr0] + dur0[pr0]+1)<=
   uSR01[pr0,pr1]*(start1[pr1] + 712(pre10[pr0,pr1]));
res10[pr0,pr1]:uSR01[pr0,pr1]*(start1[pr1] + dur1[pr1]+1)<=
   uSR01[pr0,pr1]*(start0[pr0] + 712(pre01[pr0,pr1]));
seq0[pr0]:start0[pr0] >= start0[pr0-1] + dur0[pr0-1];
seq1[pr1]:start1[pr1] >= start1[pr1-1] + dur1[pr1-1];
INTEGER
start0[pr0]
start1[pr1]
BINARY
pre01[pr0,pr1]
pre10[pr0,pr1]
BOUNDS
start0 < 356;
start1 < 356;
END
```

equation. This enables the encoding of conditional constraints[5]. In lines 26 and 27 the opposite case, the activity of order 2 is executed before the activity of order 1 is described. These constraints have to be defined for each possible pair of orders from the set of all orders. In lines 28 and 29 the sequence of activities within each order are encoded. In the lines 31 and 32 it is pointed out that the start value for each activity is an integer value. While in lines 34 and 35 the matrices `pre01` and `pre10` are defined as binary matrices. The start time of all operations, i.e., all entries in the vectors `start0` and `start1` are bounded below a fixed value.

The code in the listing has been generated by a program written for this study that generates MPL files for arbitrary number of orders.

The MPL Modeling System can transform the notation of the LP into inputs for existing solvers, like the CPLEX[6] solver. Moreover, it is shipped with the open source solver CoinMP[7], that has been used in the experiments presented here.

The modeling technique described to solve scheduling problems is similar to the modeling done by Liu and Sycara [LS96]. They model the start times of sub-tasks as variables that have to be initialized to generate a feasible schedule. In contrast to the model presented here, they use a constraint-based modeling approach, formulating a distributed constraint satisfaction problem.

Heuristics for solving the allocation

The second approach for evaluating the ABACO predictive planning abilities is an order-based scheduling heuristic. Order-based scheduling heuristics have been described in general form by Sauer [Sau93, p. 50], which is shown in Algorithm 2. Of particular interest is the design of the section of the next order that has to

Algorithm 2 Order-based scheduling heuristic

```
while exists unscheduled order do
    SELECT order
    SCHEDULE all activities for selected order
end while
```

be scheduled. To maximize the objective function it is useful to ensure that each order is finished before its due date. Thus, orders with due dates closer to the current time should be finished earlier. For that reason we choose the order with the minimal due date of all unscheduled orders. If an order has been selected all its activities are scheduled sequentially. For each activity we compute the earliest

[5]`http://www.maximalsoftware.com/support/mplfaq.html#faq4.1`, Accessed: 01/06/2010
[6]`http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`, Accessed: 01/06/2010
[7]`http://www.maximalsoftware.com/solvers/coin.html`, Accessed: 01/06/2010

start time, i.e., the start time of the order or the completion time of the preceding activity. Starting from this time we use a first fit heuristic to find a time slot on the required resource for the activity.

Note that in a distributed way, the ABACO heuristic for the initial planning can be described by this algorithm, as well. Orders are scheduled sequentially. The selection of the next order to schedule is not very sophisticated in the ABACO algorithms. The orders are scheduled sequentially in the sequence the coordinator agents reads the order from its database. This is reasonable for the ABACO system, as it relies on the assumption, that the current schedule is the product of a sequence of continuous schedule repair and schedule improvement steps. The initial scheduling is a singular event in the operation of the ABACO system. Nevertheless, we can take advantage of this fact, as we can modify the heuristic to synthesize the outcomes of the ABACO approach using a modified version of the heuristics. Of course, this is only valid for a comparison of the outcomes of the initial predictive scheduling step.

6.6. Evaluation of coordination mechanisms

Finally the last step of the ECo process is the evaluation. The context of this process step in the ECo process is given in Figure 6.10.

In the following evaluation we compare the results of the ABACO approach with other planning mechanisms, concerning their performance for the predictive planning, only. Aspects like planning under dynamics are not subject of the evaluation here, as the mechanisms used for comparison are not capable of reactive planning. Note that the ABACO approach is capable of reactive planning/scheduling, as already discussed above.

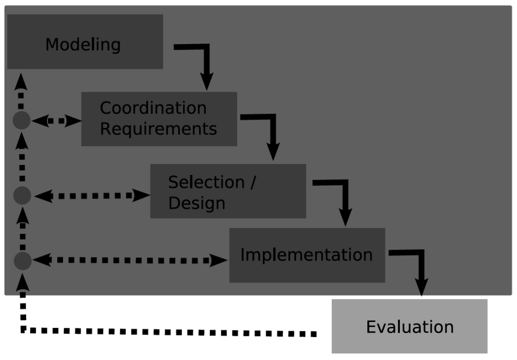


Figure 6.10.: Context of the evaluation step in the ECo process

Model simplifications

As already pointed out, we have simplified the model of the supply chain. We limit the number of participants to two. We have limited the number of possible products offered by the entire supply chain to two final products. We have eliminated variants for products. Moreover, we assume that all local planning problems are of the same type and that no alternative resources exist to fulfill activities. Thus, the resulting examples are oversimplified.

This limitations are necessary to apply conventional approaches, linear programming in particular, to be applicable. The ABACO approach do not rely on such restrictions to be applicable.

To perform comparisons between the different planning and coordination mechanisms we have designed an order generator. This order generator is used in our experiments. For each order the following elements have to be defined:

- an ID for the order,
- the product code, so which product has to be produced,
- the required quantity of the product,
- an earliest start time for the order, and
- a due date.

IDs are given sequentially increasing. The product code is chosen randomly out of the two different products. Note that for our first experiments, where we also compute optimal solutions, we only use one type of product, because the linear program presented above could not deal with alternative resource for operations, at all. The quantity requested within one orders is an arbitrary number between 1 and 5. The earliest start time for all orders is set to 0, which corresponds to a pure predictive planning. Additionally, this enables us to put extra stress to the system, as a load peak is introduced. The due date for an order is computed by the following formulae:

$$\text{start time} + (\text{quantity} * 40) + \text{rand}$$

`rand` is a random number chosen between 0 and 500. As each order contains eight steps and the longest step requires 5 time units, the quantity is multiplied with 40 which is a rough estimation of the pure execution time the order requires.

Scalability finding the optimal solution

It has already been pointed out that the scalability for computing optimal solutions is the limiting factor for the wide usage of mechanisms that ensures to finding guaranteed optimal solutions. In the following we quantify this in our experiments concerning the scalability of the optimal approach, using the CoinMP solver. Note that the scaling effects are not particular related to a specific solver, but are inherent in the problem size. For that reason, we present the average computation

orders	avg. comp. time	min. #iter.	max. #iter.	variables	constraints
1	0.00	0	0	8	8
2	0.05	2	67	144	104
3	0.54	185	3033	408	288
4	4.45	3095	66927	800	560
5	43.31	19302	501838	1320	920
6	833	970467	8893483	1968	1368
7	33790.2	23628064	318645933	2744	1904

Table 6.13.: Scalability for optimal solutions

time (in seconds) for different problem sizes. We support this measurement by the minimal and maximal number of iterations required to find the optimal solution. Moreover, the number of variables and constraints is given. All this data is summarized in Table 6.13. This data has been computed on the base of five randomly generated problem instances. Note that these results do not aim to be statistical significant. In fact we want to point out the complexity that has to be handled in finding the optimal solution. Note that we have already simplified the problem and the problem size is small in comparison to real-world problems.

For one order the computation does, of course, requires time, but the amount of time is below 0.01 seconds, which is the finest granularity the solver can measure. It becomes clear that in particular for larger instances the computational effort growth to unacceptable size.

As we are mention required computation time here for the first time, we have to provide details about the hardware used for these tests. For all tests a Lenovo T61 notebook has been used. It has an Intel Core 2 Duo 2Ghz CPU and 3GB RAM. The systems uses Windows Vista SP2 as operating system. The CoinMP solver use per default only one core. Thus, all times presented here were computed with only one core. This, of course, offers the potential to speed up computation in a near linear way[8] utilizing multicore computer architectures and solvers that can take advantage of them. This will enable to computation of additional slightly larger problems. But, as we have shown above, the complexity growth larger than linear. Thus, even for more powerful hardware architectures and elaborated solvers strict limitation exist, concerning their capability to solve larger instances, which hinders their broad application.

For up to 5 or 6 orders the resulting complexity and thus required computation time stays within acceptable boundaries in this experimental setting. For that

[8]Up to an upper limit, given by the divisibility of the overall task and the resulting interdependencies between the sub-tasks.

Instance	optimal	ABACO		due-date heuristic	
		quality	deviance in %	quality	deviance in %
Inst 1	132	35	73.48%	22	83.33%
Inst 2	393	194	50.64%	224	43%
Inst 3	360	239	33.61%	179	50.28%
Inst 4	196	-40	120.41%	73	62.76%
Inst 5	183	142	**22.40%**	43	76.50%
Inst 6	174	30	82.76	1	99.43%
Inst 7	438	303	30.82%	318	**27.40%**
Inst 8	399	281	29.57%	176	55.89%
Inst 9	386	224	41.97%	213	44.82%
Inst 10	221	52	76.47%	-45	120.36%
Mean			56.21%		66.38%

Table 6.14.: Comparison heuristics and optimal solution

reason, we use a problem size of 5 orders in the following to evaluate the performance of the predictive scheduling heuristics, both modeled using the order-based heuristic.

Assessment of order-based heuristics, and ABACO in contrast to the optimal solution

We now point out the quality of the heuristics, used by ABACO and the reference implementation presented above, in contrast to the optimal solution.

All data presented here base on 10 randomly generated test instances, each specifying 5 orders. Note that the data presented is not suitable to compare the quality of both heuristic approaches. This is done in following parts of this evaluation, where the results are based on a larger number of instances. Here we just want to point out the spread in quality between the results provided by the ABACO approach and our centralized reference heuristics in contrast to the optimal solution, found by the LP solver.

In Table 6.14 we present the results of this experiment. We compare the quality of the results, according to the aforementioned objective function, of the optimal solutions with the outcomes a predictive scheduling with the ABACO approach and the earliest due-date first heuristic. For both heuristics we compute the percentage of deviation. Percentage values greater than hundred indicate that the value of the objective function of the heuristic becomes negative, i.e., the completion time of orders is beyond their due date. What can be seen from this data

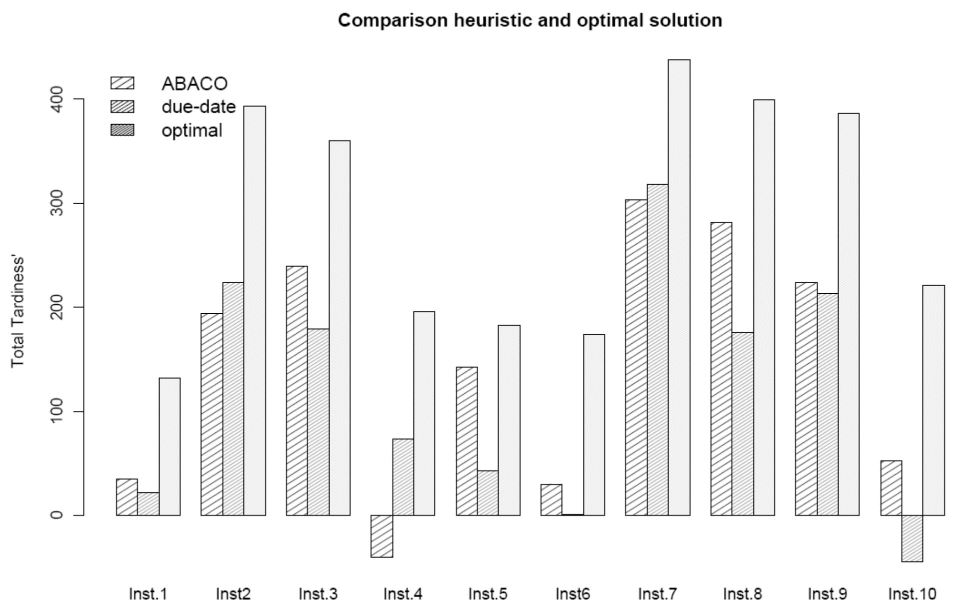


Figure 6.11.: Comparison of ABACO, due-date heuristic, and optimal solution

is, that in the mean both heuristics are more than 50% away from the optimal solution. The best case for each heuristic is highlighted in Table 6.14. For both heuristic their best case is more than 20% away from the optimal solution. To provide a more intuitive impression of these results they have been visualized in Figure 6.11. Thus, in pure predictive planning performance of both heuristics is far beyond optimal solutions. But, as pointed out in the previous section, the computation of optimal solutions is practically limited to small instances. As we are not interested in finding near optimal heuristics for this particular scheduling problem here, we use both heuristic approaches for the following experiments, as well. Keep in mind that this particular scheduling problem, even it turns out to be challenging to solve with good quality, is a strict simplification of the problems that were outlined above. Moreover, we are interested in the coordination of existing distributed planning and scheduling problems here, not in finding efficient solutions for a centralized formulation of those problems.

Relaxing the constraints on the model

In the following experiments we relax some of the restrictions that were necessary to build a linear program. In the following we allow orders to request product of type two, which can be produced by either company A or B or by both of them. Moreover, we allow to have divergent planning problems and potentially planning systems at each planning entity. Therefore, we change the assumption of how company A produces its good. We do so to introduce the characteristic that different planning authorities have to handle different planning problems and therefore use different planning systems. For that reason, we give in Table 6.15 an update of the local planning problems, in contrast to the data presented in Table 6.11. In particular the planning problem of company A has changed in a form that it is not necessary to schedule two activities at local resources but to select one resource for the execution of one activity necessary to perform the operation. Again, we use different names for resources of company A and B. This enables us to specify the scheduling problem for the centralized order-based heuristic.

Scalability of the heuristics

In this experiment we investigate the performance of the heuristic approaches under different workloads, i.e., orders to schedule. Therefore, we compute the results for hundred randomly created instances for each scenario. Thereby, a scenario specifies a certain number of orders that have to be scheduled. We created scenarios ranging from 1 up to 30 orders. For a better analysis we present the graph with our results in two parts. The first part contains the scenarios with 1 to 15 orders. The second graph shows the results with 15 and more orders. The results of the first part of our analysis can be found in Figure 6.12. Note that for one order both approaches end up with the same result, because for one order their behavior is identical. It can be clearly seen that starting from the scenario with five orders onwards the results of both heuristics diverge. Up to about a load of 8 orders the resulting objectives increase. In the scenarios from one to eight orders the addition of a new order leads to an increase of the objective function, because all orders can be finished before their due and therefore extra utility is added. From 8 to 12 orders the objective function nearly plateaus, expect an outlier for 10 orders. In these cases the additional order does not increase the objective values, because some orders are now satisfied just in time, and therefore do not increase of decrease the value of the objective function. Starting from the scenario with 13 orders the objective drops. From that point on, the addition of an extra order leads to delays in the realization of orders.

The second part of our results, ranging from 15 to 30 orders is shown in Figure 6.13. It can be observed that the objective falls nearly linear with the number of orders that have to be scheduled. In these scenarios the existing resources starts to get over utilized. Orders tend to be later finished. Note that up to 18/19 orders the objective function is positive. That is, most of the orders are finished on time and it is economically beneficial to add orders. For 20 or more orders the objective function, the total tardiness becomes negative, which indicates that jobs

Company A			
operation	activity	resource	duration
111	a	1	5
	b	3	5
114	a	3	5
	b	5	5
211	a	2	5
	b	5	5
212	a	2	5
	b	4	5
213	a	1	5
	b	2	5
214	a	4	5
	b	5	5
Company B			
112	1	6	5
	2	7	5
113	1	6	5
	2	8	5
211	1	7	2
	2	10	3
212	1	6	3
	2	9	2
213	1	7	4
	2	9	1
214	1	8	1
	2	10	5

Table 6.15.: Updated details of local production details for SC operations

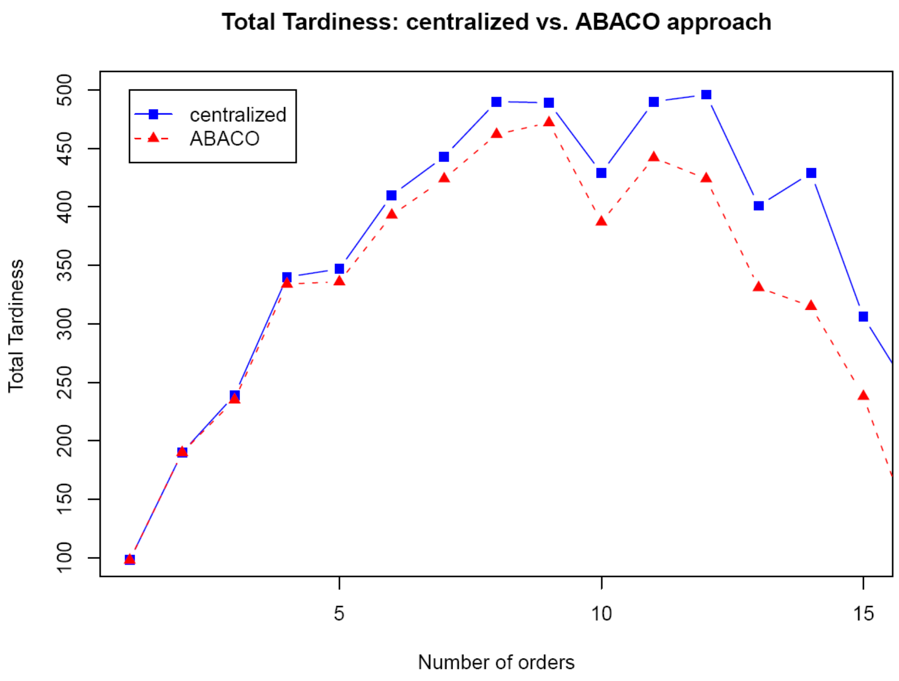


Figure 6.12.: Comparison of ABACO, due-date heuristic for 1–15 orders

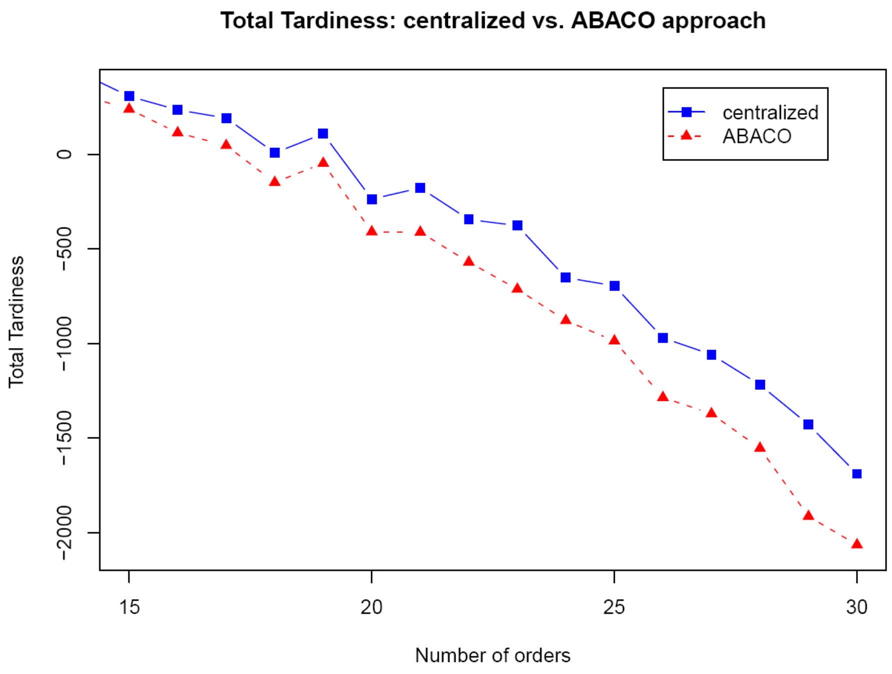


Figure 6.13.: Comparison of ABACO, due-date heuristic for 15–30 orders

#orders	heuristic	mean	std.dev.	min	max
5	edd	370.62	192.58	-447	1006
	ABACO	363.25	196.33	-588	1006
10	edd	483.80	332.88	-1199	1583
	ABACO	442.92	342.70	-1438	1530
14	edd	380.08	463.17	-2122	1804
	ABACO	298.14	483.78	-2284	1811

Table 6.16.: Comparison between heuristics

are not finished on time, anymore. This indicates an overload of the production system. The performance of the system decrease independently of which planning approach is used for the predictive scheduling.

Even if these results can already give an impression about the performance of the heuristic used in the ABACO approach and the benchmark heuristic we do not discuss there relative performance here, as the number of 100 instances does not allows serious conclusions. This comparison is done in our last experiment, described below.

Comparing the results for predictive scheduling of the ABACO approach with the order-based centralized heuristic

As pointed out above, a first trend in the results of the ABACO approach and the order-based heuristic could be seen in the previous section. In this section we directly compare the results of both heuristics. We do so, using three different scenarios with 5, 10 and 14 orders. We pick those scenarios on the base of the analysis of the results presented in Figure 6.12. At the scenario with 5 orders the spread between both approaches becomes clear. The second scenario with 10 orders it one out of the plateau, and in particular the one that differs from the other scenarios of the plateau. Finally, we choose the scenario with 14 orders, as the spread between both approaches is large between them. The results reported here are based on 10.000 different instances of each scenario. The results of this experiments are summarized in Table 6.16 and Figure 6.14. As one can see the mean values in all scenarios are smaller for the earliest due date first (edd) heuristic. But the statistical spread between both approach is very similar, in terms of standard deviation, minimal and maximal values. This relativize the quality of the edd heuristic in comparison with the heuristic used in the ABACO approach. This conclusion is supported by the box blots of these scenarios, where both heuristic approaches are compared with each other. The box plots are shown in Figure

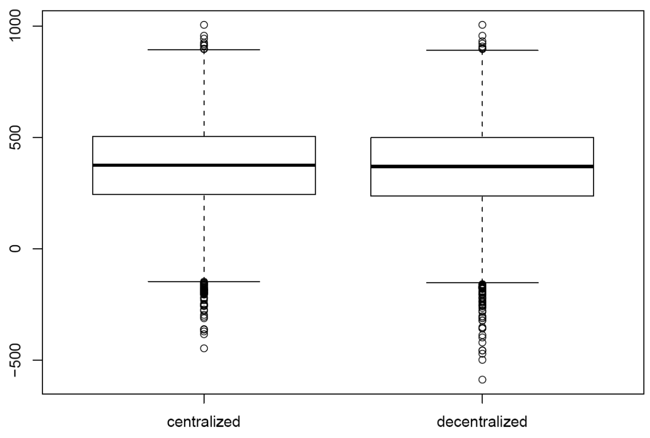
6.14. As one can see in these box plots the outcome of both heuristics does not differ significantly, which is also supported by the results of t–test, that indicate no significant difference, as well.

In summary, two issues can be stated about the performance of the ABACO heuristic. First, in comparison to the optimal solution the heuristic used within the ABACO approach is outperformed by numbers. Second, in comparison to an edd order-based heuristic no significant differences can be found. Even if the mean performance is slightly worse, this cannot be verified for the entire data set, as the statistical spread of both approaches is wide. Of course, one can argue that more sophisticated centralized optimization strategies, like metaheuristics for instance, could find better solutions and thus outperform the performance of the ABACO approach. But the ABACO approach has two major advantages in favor to these centralized scheduling algorithms. First, it satisfy the coordination requirements, and thus can achieve these results respecting the existence of planning systems in each company. Second, the ABACO approach has mechanisms for reactive planning, which at least simple heuristics do not have, and thus need to perform a complete replanning. These abilities include techniques for direct reaction to events that invalidate the current global plan, and additionally enables the participating companies to try to search for local improvements, which increase plan quality over time. Plan improvement is an important supplementing technique, as the overall plan quality tends to decrease by a sequence of plan repair steps, performed by the reactive scheduling methodology.

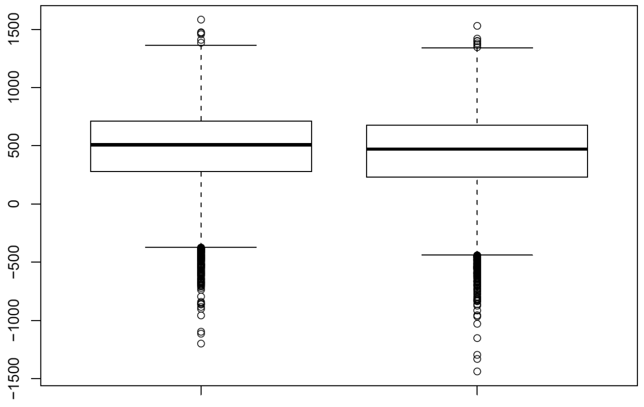
6.7. Criticism of the ECo-CoPS approach

The modeling of the coordination problems, as a distributed planning problem, containing local planning problems in itself, can become a complex task. Especially because it is not clear in the beginning which concepts are required to formalized each and every coordination requirement. Thus, it might be useful to iteratively refine the model while trying to formalize the coordination requirements. This procedure goes perfectly in line with the ECo process, which propose an iterative process model, for its execution.

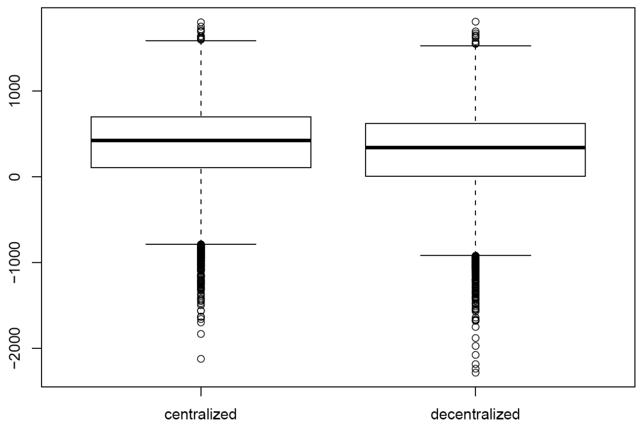
The formalization of necessary criteria for the applicability of a coordination mechanism has not been discussed in research, so far. It facilitates pointing out clearly the requirements of a given problem, and moreover, it is possible to use coordination requirements for the explanation of design decisions of coordination mechanisms. Note that these coordination requirements go far beyond purely technical aspects, especially the non-functional requirements, like information hiding, maintaining local autonomy, and preserving existing planning systems, are re-



(a) 5 orders



(b) 10 orders



(c) 14 orders

Figure 6.14.: Box plots for instances with 5, 10, and 14 orders

quirements that are motivated by real applications with economic constraints. The capturing of non-functional coordination requirements has been discussed in this case study. By applying the ECo process the selection of a coordination mechanism can be guided by those constraints.

In this case study we identified an existing coordination mechanism for the coordination, and consequently, an existing implementation. For that reason the CoPS framework has not been used here. Moreover, a solver for linear programs was used to find the optimal solution for a reduced problem and another centralized heuristic planner has been used, as well.

In the quantitative evaluation we give strong indications that both heuristic approaches (the centralized one, and the ABACO predictive planning) are far away from optimal solutions, but only require small fractions of computation time. In a more detailed analysis, we found no statistical significant quality decrease using the centralized heuristic and the ABACO approach for the initial planning. But, in contrast to the centralized planning heuristic, the ABACO approach satisfies all coordination requirements that have been formulated. Moreover, the ABACO approach offers specific additional features for reactive planning and plan improvement facilities.

For the selection of a coordination mechanism the ECo process has reconfirmed the decision to construct a new coordination approach in our previous work. The ABACO approach is identified as the only existing coordination mechanism, that satisfy all coordination requirements. A re-implementation is not done here, as the original implementation can be used in the study.

7. Conclusion and Perspectives

Complex planning and decision problems are often solved by decomposing the overall problem into smaller, and therefore, often simpler sub-problems. These sub-problems are solved, often independent from each other, and the resulting partial solutions are synthesized to a global solution for the overall problem. This strategy has been implemented in industry and academia, as well. Complex problems like the container terminal management problem are decomposed into *simpler* problems like the berth allocation, crane scheduling, storage space allocation, and the transportation problem, for instance. Often these sub-problems turn out to be very complex, as well. In consequence, these isolated problems are addressed by researchers as a topic of research by itself. It was pointed out in our discussion about dependencies between planning systems (see Section 2.2.4), that the sub-problems can have various interdependencies. Thus, the computed partial plans cannot simply be combined to construct a feasible global plan. For that reason, coordination mechanisms have to be put in place to ensure that the resulting partial plans can be executed together and an acceptable, or ideally good to optimal, overall performance is ensured.

We have addressed here the coordination on the operative level. The decomposition of the overall problem, the methods for solving partial problems, and the organizational formation of planning entities are assumed to be given. Therefore, the coordination mechanisms should only effect decisions on the operative level. In Section 3.1 we have outlined a number of existing coordination approaches that have been proposed mainly by researchers from the field of DAI, addressing the problem of coordination of planning entities on the operative level. The issue of coordination is a centerpiece in DAI research. Consequently, a variety of different approaches have been presented. Each addressing a certain aspect of the coordination problem, or designed for a specific coordination problem.

Findings of the study

We summarize the main findings of this work here. In this thesis

- a research gap for the selection of coordination mechanisms for existing autonomous planning systems has been identified,

- the ECo process has been designed to bridge this gap,
- the feasibility of the ECo process has been proven by applying it in two case studies,
- providing the CoPS process and implementing the CoPS framework, the implementation step of the ECo process is detailed and supported,
- in the case studies the ECo-CoPS approach has proven to be advantageous for coordination problems in the logistic domain.

To profit from existing research concerning the coordination of autonomous agents, it is promising to re-use existing methodologies. A large body of research has been documented in the literature, and numerous coordination techniques for agent-based systems have been proposed. As there exists no suitable classification of existing mechanisms, the identification of applicable mechanisms for a given situation is a complex and time consuming task. We have presented a survey in the field of agent-oriented software engineering (see Section 3.2) to identify suitable mechanisms for the identification and re-use of existing concepts, in particular coordination mechanisms. A major result of our survey is that there is rarely work done dealing with re-use of methodologies in AOSE. The selection of coordination mechanisms is a field that is currently not covered. We pointed out this research gap in Section 3.3 and the necessity for means to bridge this gap. Thereby, we focus on the coordination of existing distributed planning systems, and emphasize the importance of the applicability of the approach in the given context.

A first, but significant, step towards our goal is the identification of key characteristics of coordination problems. We extracted those characteristics from the examples used in our study. We pointed out six different characteristics of coordination problems (see Section 4.1). These characteristics are:

- the existence of an allocation problem,
- the existence of comparable local objective functions,
- the diversification of existing planning systems,
- the existence of a common overall objective function,
- the necessity of information hiding, and
- the existence of cyclic dependencies.

We have classified the presented coordination mechanisms according to the characteristics they require. In our case studies the classification turns out to be very efficient in the selection process. It enabled us to focus the search on a reduced set of candidate approaches. In the following detailed analysis, the qualitative evaluation, the coordination mechanisms were analyzed according to their compliance with the coordination requirements. This selection process is open in the sense that the outcome can be either the identification of existing mechanisms, with or without existing implementation, or the finding that no appropriate mechanism exists.

To enable efficient identification of suitable coordination mechanisms we presented the ECo process, that guides the identification and selection process (see Section 4.2). The ECo process contains five steps that can be executed in an iterative process model shown in Figure 7.1. The steps of the ECo process are:

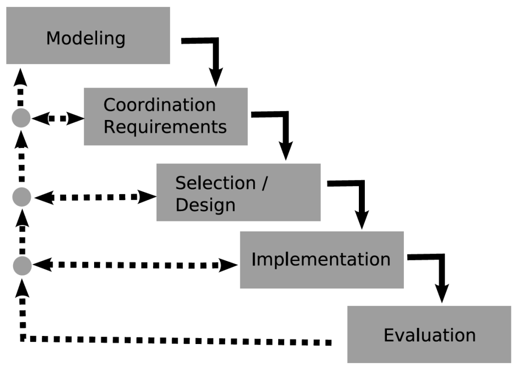


Figure 7.1.: The ECo process

1. modeling the scenario,
2. defining coordination requirements,
3. identifying suitable coordination mechanisms,
4. implementing coordination mechanisms, and
5. evaluating coordination mechanisms.

The introduction of coordination requirements is a significant step towards a more explicit formalization capturing what a coordination mechanism has to fulfill to be applicable in a given situation. If coordination mechanisms are classified according to requirements they fulfill, this will facilitate the reuse, and leverage the advantages of requirements. The selection out of classified mechanisms becomes simpler as detailed analyzing processes can be omitted. Coordination requirements also facilitate the documentation of design decisions that were made during the design and implementation phase.

A coordination mechanism is applicable if it satisfies all functional and non-functional coordination requirements. The evaluation if these requirements are satisfied is called qualitative evaluation. As the requirements are formalized on a sound model, the qualitative evaluation can be done in a formal way. The result of the qualitative evaluation is a set of mechanisms that are applicable in the given situation, as they satisfy all coordination requirements. Note that this result set can be empty, as well. Moreover, the qualitative evaluation does not provide any indication of the quality of the coordinated plans, except the ones encoded in the requirements, which is typically feasibility of the joint execution of the plans.

To support the implementation step, the CoPS process is proposed (see Section 4.3). The CoPS process structures decision making that has to be made during the implementation of a coordination mechanism. It spawns from decisions made on the layer of the overall network, like the selection of conversation protocols, to decisions that have to be made at each participating planning entity, like the definition of conversation policies or the adaptation of existing planning systems. To ease the implementation process, the CoPS framework has been introduced. Within this framework an infrastructure is implemented. It facilitates fast implementation/prototyping of coordination mechanisms. By enabling to prototype coordination mechanisms the CoPS approach facilitates the evaluation of mechanisms on a quantitative base. The CoPS framework can also speed up the implementation of new coordination mechanisms.

The CoPS framework is currently not intended for commercial implementations. It has deficits concerning aspects like failover abilities and adapters to standard software systems that are necessary for operating in a productive environment.

In the final step of the ECo process a quantitative evaluation is suggested to evaluate the performance of the selected coordination mechanisms, which enables the final judgement which mechanism should be used. This evaluation uses implemented systems. Thus, the evaluation can be done using real-world like data in a real-world like environment.

The ECo-process supported by the CoPS process and the CoPS framework form the ECo-CoPS approach that is introduced in this thesis. In Chapter 5 and Chapter 6 we applied the ECo-CoPS approach to two case studies from the field of production and logistics. In these two case studies we have validated the ECo process by pointing out its applicability, and found indications that this process adds additional value to the selection process of coordination mechanisms. Our experience with the ECo-CoPS approach have been discussed in the previous chapters. We can summarize them as follows. The ECo and CoPS process facilitate a structured procedure in the identification, selection, and implementation phase of suitable coordination mechanisms. To draw conclusions about the CoPS framework more case studies are needed.

Aspects that have been identified during our case studies, and that might complicate efficient usage of the ECo-CoPS approach are:

- The modeling of the coordination problems can become complex, and multiple iterations between modeling and formulation of coordination requirements can be necessary. Therefore, it might be useful to investigate different types of modeling techniques. Also, a dedicated sub-process for the modeling step might be valuable.

- The CoPS framework requires more evaluation, and thereby has to be evolved.

It can be argued that the coordination mechanisms discussed in this thesis do not favor optimal outcomes on the global level, which is, for instance, the goal of collaborative planning, presented in Section 3.1.1, or other optimization approaches. This criticism is correct. As we have stated, we address the coordination of existing planning systems, and have even stated in the second case study that local autonomy has to be preserved. This strictly limits the ability of the overall system to come to an optimal global solution. This has been shown in particular in the second case study, where a comparison with an optimal solution has been performed. The loss of global optimality is a direct consequence of our approach to restrict ourselves to the coordination of existing planning systems in existing autonomous planning entities. We did so, because planning systems have been used in real world applications for decades, and are often well established. These software systems, and the organizational structures of companies are fixed from the perspective of the operational level that is addressed in this research. Thus, this limitation results from assumptions we have made, because we are convinced that they will allow us to model real-world like scenarios more accurate.

What also has not been addressed in this study is the empirical evaluation of the ECo process. The evaluation of a process is a social science task, as well. It is necessary to apply the process a number of times, and each process execution has to be observed, and criticism has to be recorded. An experimental setting would be to have two equally skilled groups facing the same problem, and then let one group use the ECo process while the other group does not. The second group is the control group. Moreover, the context, like given deadlines or given technology, has to be taken into account, as this can effect the quality of the outcome of the process, as well. These experiments have to be replicated to draw sound conclusions. Therefore, it is necessary that each participant of such a study can participate in exactly one experiment. Replications are also required for every application domain that is evaluated.

Perspectives for future research

The research gap identified in this thesis is covered by the proposed ECo-CoPS approach. The results of this thesis can be an initial pre-requisite for a number of subsequent research projects. We outline some aspects here, that should and will be investigated in future research.

As already mentioned, the number of case studies applying the ECo-CoPS approach needs to be increased to allow for more reliable evaluation results, concerning the additional value the ECo-CoPS approach offers in contrast to an unstructured process. Therefore, it is necessary to apply the ECo-CoPS approach, and contrasting conversational approach, to various domains, and, ideally, the process should be performed by different developers. A first candidate for the next case study is the coordination of planning systems in container terminal management, which has been used in this thesis as an example, e.g., planning systems for this domain have already been provided by Leif Meier [Mei08]. The ECo-CoPS approach has been designed to be generally applicable. However, if a variety of case studies exist, it could become useful to analyze if domains exist that are particular well suited for the application of the ECo-CoPS approach, and if application areas exist that require more specialized or enhanced versions of the process or additional guidance.

Another aspect that could be used in following case studies is the investigation of different modeling techniques for the coordination problems. As already pointed out, it can become necessary to iterate between modeling and formulation of coordination requirements, and modeling can become a complex task. It might be useful to investigate modeling techniques that facilitate on the one hand a formally grounded description of the scenario, and, on the other hand, offer additional value during the implementation, and can therefore foster an implementation phase. Modeling with UML profiles and the object constraint language (OCL) might be a first starting point for alternative techniques for modeling.

In the course of the design of conversation protocols, and in particular for the definition of conversation policies it turned out that the existing work is often not sufficient. This is especially true, if the agent's behavior should be specified by the humans they have to represent, and, thereby, adapt the negotiation strategies of the humans. Specifying agent's behavior is often strictly tied to programming code the agent can use, or that defines the agent's behavior. Thus, the modeling of conversations, and conversation policies is still an open research issue. Even if interaction protocols are nowadays widely accepted and standardized, there rarely exist any techniques for modeling conversation policies or strategies for consistency among parallel negotiations performed by an agent with multiple other agents at the same time.

In the ECo process we emphasized the importance of coordination requirements. They can be used to specify characteristics of the current context of the coordination problem, and formalize criteria that have to be regarded by coordination mechanisms to be applicable. Moreover, they offer additional value. Existing coordination mechanisms can be classified according to a fixed set of coordination requirements, and, therefore, form a catalog of coordination mechanisms that can be used to foster the identification of candidates for a new coordination problem. Thus, coordination requirements build a new scheme for the classification of coordination mechanisms that can offer practical value by encouraging and easing re-use of existing mechanisms.

In this thesis, we have strictly limited our focus on the coordination of activities on the operational level. The tactical and strategy decision level have been excluded from this study. An example of a tactical decision might be to decide if, and when, to enter/leave a network. A decision on the strategic level is the decomposition of the overall planning problem into sub-problems and planning entities. This decomposition is an interesting topic of research, as well. Therefore, we investigate in a subsequent research project the effects of different decompositions on the achievable overall quality of coordinated solutions.

We are going to model the overall problem as a distributed constraint optimization problem with privacy issues. A constraint optimization problem (COP) is a constraint satisfaction problem with an additional objective function for solutions that has to be maximized. This optimization problem can be distributed. Different solvers are responsible for distinct subsets of variables. Privacy issues can be added if variables, domains, and constraints that are located at exactly one solver are only known to this particular solver. If a constraint involving variables located at two or more different solvers exists, these variables and the constraints are semi-private, as are all participating solvers, and only those are aware of the existence of the variables within this constraint. Distributed constraint optimization problems with privacy issues can be used to model the coordination of planning systems. Each local planning system can be modeled as a constraint optimization problem in itself, and additional constraints can be introduced representing the interdependencies between different planning systems. Thus, it is possible to use constraint optimization techniques to identify optimally coordinated solutions, which can be used as benchmarks. This large COP would be hard to solve optimally. This enable a comparison of results of a centralized problem with results of different decompositions. Of particular interest is the identification of decomposition techniques. This research could give valuable insight on the effects of distribution techniques on achievable solution quality.

Bibliography

[ABTV04] Dietrich Adam, Klaus Backhaus, Ulrich W. Thonemann, and Markus Voeth. *Allgemeine Betriebswirtschaftslehre - Koordination betrieblicher Entscheidungen.* Springer, Berlin, 3rd edition, 2004.

[ACF+98] Rachid Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An architecture for autonomy. *International Journal of Robotics Research*, 17(4), 1998.

[AFHS95] Oksana Arnold, Wolfgang Faisst, Martina Härtling, and Pascal Sieber. Virtuelle Unternehmen als Unternehmenstyp der Zukunft? *HMD*, 185:8 – 23, 1995.

[AGO08] Marlene Arangú, Antonio Garrido, and Eva Onaindia. A general technique for plan repair. In *Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence*, volume 1, pages 515–518, Dayton, Ohio,USA, 2008. IEEE Computer Society.

[AM09] Karl Johan Aström and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers.* Princeton University Press, Princeton and Oxford, 2009.

[AO94] Krzysztof R. Apt and Ernst-Rüdiger Olderog. *Programmverifikation.* Springer, Berlin, 1994.

[BCG+98] Frances M. T. Brazier, Frank Cornelissen, Rune Gustavsson, Catholijn M. Jonker, Olle Lindeberg, Bianca Polak, and Jan Treur. Compositional design and verification of a multi-agent system for one-to-many negotiation. In *Proceedings of the Third International Conference on Multi-Agent Systems, ICMAS'98*, pages 49 – 56. IEEE Computer Society Press, 1998.

[BCGZ01] Nadia Busi, Paolo Ciancarini, Roberto Gorrieri, and Gianluigi Zavattaro. Models for coordinating agents: a guided tour. In Andrea Omicini, Robert Tolksdorf, Gerhard Weiss, and Franco

Zambonelli, editors, *Coordination for Internet Agents: Models,Technologies, and Applications*, pages 6–24. Springer, 2001.

[BEY98] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis.* Cambride University Press, Cambridge, 1998.

[BF95] Mihai Barbuceanu and Mark S. Fox. Cool: A language for describing coordination in multi agent systems. In *Proceedings of the First International Conference on Multiagent Systems (ICMAS95)*, pages 17 – 24. AAAI Press, 1995.

[BFJ$^+$04] Jeffrey M. Bradshaw, Paul J. Feltovich, Hyuckchul Jung, Shirniwas Kuklarni, William Taysom, and Andrzej Uszok. Dimensions of adjustable autonomy and mixed-initiative-interaction. In Matthias Nickles, Michael Rovatsos, and Gerhard Weiss, editors, *AUTONOMY 2003 : International Workshop on computational autonomy*, volume 2969 of *LNAI*, pages 17 – 39, Melbourne, 2004. Springer, Berlin.

[BGZ04] Federico Bergenti, Marie-Pierre Gleizes, and Franco Zambonelli, editors. *Methodologies and Software Engineering for agent systems: The Agent-oriented softwre Engineering Handbook.* Multiagent systems, artificial societies, and simulated organizations. Kluwer Academic Publishers, Bosten, 2004.

[BH09] Nils Bulling and Koen V. Hindriks. Towards a verification framework for communicating rational agents. In Lars Braubach, Wiebe van der Hoek, Paolo Petta, and Alexander Pokahr, editors, *Multiagent System Technologies (MATES 2009)*, Lecture Notes in Artificial Intelligence (LNAI), pages 177 – 182, Hamburg, 2009. Springer.

[BHW07] Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge. *Programming Multi-Agent Systems in AgentSpeak using Jason.* Wiley Series in Agent Technology. John Wiley & Sons Ltd., 2007.

[BJW03] Stefan Bussmann, Nick R. Jennings, and Michael Wooldridge. Reuse of interaction protocols for agent-based control applications. In Fausto Giunchiglia, James Odell, and Gerhard Weiss, editors, *Agent-Oriented Software Engineering III Proc. of the Third International Workshop, AOSE 2002*, volume 2585 of *Lecture Notes in Computer Science*, pages 73 – 87, Bologna, Italy, 2003. Springer.

[BM92] Stefan Bussmann and H. Jürgen Müller. A negotiation framework for cooperating agents. In S.M.Deen, editor, *Proc. of the CKBS-SIG (CKBS'92)*, pages 1 –17, DAKE Centre, Univ. of Keele, 1992.

[BMR$^+$02] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *A System of Patterns: Pattern-Oriented Software Architecture*, volume 1 of *Wiley Series in Sotware Design Patterns*. John Wiley & Sons Ldt., Chichester, 2002.

[BMRL03] Paolo Busetta, Mattia Merzi, Silvia Rossi, and Franc¸ois Legras. Realtime role coordination for ambient intelligence. In *Workshop Representations and Approaches for TimeCritical Decentralized Resource/Role/Task Allocation at AAMAS03*, Melbourne, Australia, 2003.

[BO00] Robert W. Brennan and William O. A simulation test-bed to evaluate multi-agent control of manufacturing systems. In *WSC '00: Proceedings of the 32nd conference on Winter simulation*, pages 1747–1756, Orlando, Florida, 2000. Society for Computer Simulation International.

[BPJ03] Claudio Bartolini, Chris Preist, and Nick R. Jennings. Architecting for reuse: A software framework for automated negotiation. In John Mylopoulos, Michael Winikoff, and Nick R. Jennings, editors, *Agent-Oriented Software Engineering III Proc. of the Third International Workshop, AOSE 2002*, volume 2585, pages 88 – 100, Bologna, Italy, 2003. Springer.

[BR02] Federico Bergenti and Alessandro Ricci. Three approaches to the coordination of multiagent systems. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, pages 367–372, Madrid, Spain, 2002. ACM Press.

[Bra87] Michael E. Bratman. *Intention, plans, and practical reason.* Harvard University Press, Cambridge, Mass, 1987.

[Bro86] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14 – 23, 1986.

[BTF97] Mihai Barbuceanu, Rune Teigen, and Mark S. Fox. Agent based design and simulation of supply chain systems. In *Proc. of the 6th Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises (WET-ICE '97)*, pages 36 – 42, 1997.

[CBM$^+$99] Sergio Cavalieri, Luc Bongaerts, Marco Macchi, Marco Taisch, and Jo Weyns. A benchmark framework for manufacturing control. In *Second International Workshop on Intelligent Manufacturing Systems*, pages 225 – 236, Leuven, Belgium, 1999.

[CCG$^+$04] Giovanni Caire, Wim Coulier, Francisco Garijo, Jorge J. Gómez-Sanz, Juan Pavón, Paul Kearney, and Philippe Massonet. The message methodology. In Federico Bergenti, Marie-Pierre Gleizes, and Franco Zambonelli, editors, *Methodologies and Software Engineering for agent systems: The Agent-oriented softwre Engineering Handbook*, Multiagent systems, artificial societies, and simulated organizations, pages 177 – 194. Kluwer Academic Publishers, Bosten, 2004.

[CG99] Kathleen M. Carley and Les Gasser. Computational organization theory. In Gerhard Weiss, editor, *Multiagent Systems: a modern approach to distributed artificial intelligence*, pages 299 – 330. MIT Press, 1999.

[CG01] Hans Corsten and Ralf Gössinger. Unternehmensnetzwerke: Grundlagen - Ausgestaltungsformen - Instrumente. Schriften zum Produktionsmanagement 38, Lehrstuhl für Produktionswirtschaft Universität Kaiserslautern, Februar 2001 2001.

[CGMT00] Sergio Cavalieri, Marco Garetti, Marco Macchi, and Marco Taisch. An experimental benchmarking of two multi-agent architectures for production scheduling and control. *Computers in Industry*, 43:139 – 152, 2000.

[CJ96] David Cockburn and Nick R. Jennings. Archon: A distributed artificial intelligence system for industrial applications. In G.M.P. O'Hare and Nick R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, Sixth-Generation Computer Technology Series, pages 319 – 344. John Wiley & Sons, New York, 1996.

[CK05] William Cushing and Subbarao Kambhampati. Replanning: a new perspective. In *Poster Program. In Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS-05)*, Montery, 2005.

[CLM04] J.-F. Cordeau, G. Laporte, and A .Mercier. Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *Journal of the Operational Research Society*, 55(5):542–546, 2004.

[Com07] FIPA Modeling Technical Committee. Auml web site, 2007. `http://www.auml.org/`; Accessed: 02/04/10.

[CTMG98] John Collins, Maxim Tsvetovat, Bamshad Mobasher, and Maria Gini. Magnet: A multi-agent contracting system for plan execution. In *Proc. of the Workshop on Artificial Intelligence and Manufacturing (SIGMAN'98)*, pages 63–68, Albequerque, New Mexico, 1998. AAAI Press.

[CW01] Paolo Ciancarini and Michael Wooldridge, editors. *Agent-Oriented Software Engineering, First International Workshop AOSE 2000,*, volume 1957 of *Lecture Notes in Computer Science*. Springer, Berlin, 2001.

[CWDH00] Graham Coates, Robert Ian Whitfield, Alex H. B. Duffy, and Bill Hills. Coordination approaches and systems - part ii: An operational perspective. *Research in Engineering Design*, 12:73 – 89, 2000.

[CWHC06] Marcelo Cataldo, Patrick A. Wagstrom, James D. Herbsleb, and Kathleen M. Carley1. Identification of coordination requirements: Implications for the design of collaboration and awareness tools. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 353–362, Banff, Alberta, Canada, 2006. ACM Press.

[Dec95] Keith S. Decker. *Environment Centered Analysis and Design of Coordination Mechanisms.* Phd thesis, University of Massachusetts, 1995.

[Dec96] Keith S. Decker. Distributed artificial intelligence testbeds. In G.M.P. O'Hare and Nick R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, Sixth-Generation Computer Technologies, pages 119 – 138. John Wiley & Sons, New York, 1996.

[DGSS96] Jürgen Dorn, Mario Girsch, Günther Skele, and Wolfgang Slany. Comparison of iterative improvement techniques for schedule optimization. *EJORS European Journal of Operational Research*, 94(2):349–361, 1996.

[DHK02] Ralph Depke, Reiko Heckel, and Jochen Malte Küster. Formal Agent-Oriented Modeling with UML and Graph Transformation. *Science of Computer Programming*, 44(2):229–252, 2002.

[DJT01] Mehdi Dastani, Catholijn M. Jonker, and Jan Treur. A requirement specification language for configuration dynamics of multi-agent systems. In Michael Wooldridge, Gerhard Weiss, and Paolo Ciancarini, editors, *Proceedings of Second International Workshop on Agent-Oriented Software Engineering II (AOSE 2001)*, volume 2222 of *Lecture Notes In Computer Science;*, pages 169 – 187, Montreal, Canada, 2001. Springer.

[DL87] Edmund H. Durfee and Victor Lesser. Planning coordinated actions in dynamic domains. Technical Report UM-CS-1987-130, Computer Science Department University of Massachusetts, 1987.

[DL92] Keith S. Decker and Victor Lesser. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems*, 1:319–346, 1992.

[DL93] Keith S. Decker and Victor Lesser. Analyzing a quantitative coordination relationship. *Group Decision and Negotiation*, 2(3):195–217, 1993.

[DL00] Keith S. Decker and Jinjiang Li. Coordinating Mutually Exclusive Resources using GPGP. *Autonomous Agents and Multi-Agent Systems*, 3(2):133 – 157, 2000.

[DMLS99] Sophie D'Amours, Benoit Montreuil, Pierre Lafrancois, and Fancois Soumis. Networked manufacturing: The impact of information sharing. *International Journal of Production Economics*, 58:63 – 79, 1999.

[Dor04] Jürgen Dorn. Evaluating reactive scheduling systems. In *IAT '04: Proceedings of the Intelligent Agent Technology, IEEE/WIC/ACM International Conference on (IAT'04)*, pages 458–461. IEEE Computer Society, 2004.

[DS83] R. Davis and R. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20:63 – 109, 1983.

[Dud04] Gregor Dudek. *Collaborative Planning in Supply Chains: A negotiation-based approach.* Lecture Notes in Economics and Mathematical Systems. Springer, Berlin, 2004.

[Dur99] Edmund H. Durfee. Distributed problem solving and planning. In Gerhardt Weiß, editor, *Multiagent Systems: a modern approach to distributed artificial intelligence*, pages 121 – 164. MIT Press, 1999.

[Dyc90] Harald Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research (EJOR)*, 44:145 – 159, 1990.

[EHNO08] Thomas Epping, Winfried Hochstättler, Robert Nickel, and Peter Oertel. Order sequencing in the automobile industry. In Lars Mönch and Giselher Pankratz, editors, *Intelligente Systeme zur Entscheidungsunterstützung,Teilkonferenz der Multikonferenz Wirtschaftsinformatik*, pages 49–64, München, 2008. SCS Publishing House.

[Ele04] Thomas Elendner. *Winner Determination in Combinatorial Auctions: Market-based Scheduling.* Logos Verlag, Berlin, 2004.

[ETJ04] Cora Beatriz Excelente-Toledo and Nick R. Jennings. The dynamic selection of coordination mechanisms. *Autonomous Agents and Multi-Agent Sytems*, 9:55 – 85, 2004.

[FBH07] Helmut Frank, Annette Bobrik, and Nico Haarländer. Vorgehensmodell. In Hermann Krallmann, Marten Schönherr, and Matthias Trier, editors, *Systemanalyse im Unternehmen: Prozessorientierte Methoden der Wirtschaftsinformatik*, pages 135–186. Oldenbourg, München, 5th. edition, 2007.

[Fer99] Jacques Ferber. *Multi-Agent Systems: An Introduction to Distributed Artifcial Intelligence.* Pearson Edication Ltd., London, 1999.

[FFGSP06] Rubén Fuentes-Fernández, Jorge J. Gómez-Sanz, and Juan Pavón. Requirements elicitation for agent-based applications. In Jörg P. Müller and Franco Zambonelli, editors, *Agent-Oriented Software Engineering VI: Proc. of the 6th InternationalWorkshop, AOSE 2005*, volume 3950 of *Lecture Notes in Computer Science*, pages 40 – 53, Utrecht, Netherlands, 2006. Springer.

[FFMM93] Tim Finin, Rich Fritzson, Don McKay, and Robin McEntire. Kqml - a language and protocol for knowledge and information exchange. In *Int. Conf. on Building and Sharing of Very Large-Scale Knowledge Bases*, Tokyo, 1993.

[FGLS06] Maria Fox, Alfonso Gerevini, Derek Long, and Ivan Serina. Plan stability: Replanning versus plan repair. In Derek Long, Stephen F. Smith, Daniel Borrajo, and Lee McCluskey, editors, *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS 2006)*, pages 212–221, Cumbria, UK, 2006. AAAI Press.

[FGR05] Giancarlo Fortino, Alfred Garro, and Wilma Russo. An integrated approach for the development and validation of multi-agent systems. *Computer Systems Science and Engineering*, 4:259 – 271, 2005.

[FGSP04] Rubén Fuentes, Jorge J. Gómez-Sanz, and Juan Pavón. Verification and validation techniques for multi-agent systems. *Upgrade, Council of European Informatics Societies*, 5(4):15– 19, 2004.

[FHK$^+$07] Truls Flatberg, Geir Hasle, Oddvar Kloster, Eivind J. Nilssen, and Atle Riise. Dynamic and stochastic vehicle routing in practice. In Vasileios Zeimpekis, Christos D. Tarantilis, George M. Giaglis, and Ioannis Minis, editors, *Dynamic Fleet Management: Concepts, Systems, Algorithms & Case Studies*, Operations Research/Computer Science Interfaces Series, pages 41 – 63. Springer, 2007.

[FIP01a] Foundations for Intelligent Physical Agents FIPA. Fipa agent software integration specification, 2001. `http://www.fipa.org/specs/fipa00079/index.html`; Accessed: 02/04/10.

[FIP01b] Foundations for Intelligent Physical Agents FIPA. Fipa dutch auction interaction protocol specification, 2001. `http://www.fipa.org/specs/fipa00032/index.html`; Accessed: 02/04/10.

[FIP01c] Foundations for Intelligent Physical Agents FIPA. Fipa english auction interaction protocol specification, 2001. `http://www.fipa.org/specs/fipa00031/index.html`; Accessed: 02/04/10.

[FIP02a] Foundations for Intelligent Physical Agents FIPA. FIPA Abstract Architecture Specification, 2002. `http://www.fipa.org/specs/fipa00001/index.html`; Accessed: 02/04/10.

[FIP02b] Foundations for Intelligent Physical Agents FIPA. Fipa acl message representation in bit-efficient specification, 2002. `http://www.fipa.org/specs/fipa00069/index.html`; Accessed: 02/04/10.

[FIP02c] Foundations for Intelligent Physical Agents FIPA. Fipa acl message structure specification, 2002. `http://www.fipa.org/specs/fipa00061/index.html`; Accessed: 02/04/10.

[FIP02d] Foundations for Intelligent Physical Agents FIPA. Fipa agent message transport service specification, 2002. `http://www.fipa.org/specs/fipa00067/index.html`; Accessed: 02/04/10.

[FIP02e] Foundations for Intelligent Physical Agents FIPA. Fipa brokering interaction protocol specification, 2002. `http://www.fipa.org/specs/fipa00033/index.html`; Accessed: 02/04/10.

[FIP02f] Foundations for Intelligent Physical Agents FIPA. Fipa contract net interaction protocol specification, 2002. `http://www.fipa.org/specs/fipa00029/index.html`; Accessed: 02/04/10.

[FIP02g] Foundations for Intelligent Physical Agents FIPA. Fipa iterated contract net interaction protocol specification, 2002. `http://www.fipa.org/specs/fipa00030/index.html`; Accessed: 02/04/10.

[FIP02h] Foundations for Intelligent Physical Agents FIPA. Fipa nomadic application support specification, 2002. `http://www.fipa.org/specs/fipa00014/index.html`; Accessed: 02/04/10.

[FIP02i] Foundations for Intelligent Physical Agents FIPA. Fipa propose interaction protocol specification, 2002. `http://www.fipa.org/specs/fipa00036/index.html`; Accessed: 02/04/10.

[FIP02j] Foundations for Intelligent Physical Agents FIPA. Fipa query interaction protocol specification, 2002. `http://www.fipa.org/specs/fipa00027/index.html`; Accessed: 02/04/10.

[FIP02k] Foundations for Intelligent Physical Agents FIPA. Fipa recruiting interaction protocol specification, 2002. `http://www.fipa.org/specs/fipa00034/index.html`; Accessed: 02/04/10.

[FIP02l] Foundations for Intelligent Physical Agents FIPA. Fipa request interaction protocol specification, 2002. `http://www.fipa.org/specs/fipa00026/index.html`; Accessed: 02/04/10.

[FIP02m] Foundations for Intelligent Physical Agents FIPA. Fipa request when interaction protocol specification, 2002. `http://www.fipa.org/specs/fipa00028/index.html`; Accessed: 02/04/10.

[FIP02n] Foundations for Intelligent Physical Agents FIPA. Fipa sl content language specification, 2002. `http://www.fipa.org/specs/fipa00008/index.html`; Accessed: 02/04/10.

[FIP02o] Foundations for Intelligent Physical Agents FIPA. Fipa subscribe interaction protocol specification, 2002. `http://www.fipa.org/specs/fipa00035/index.html`; Accessed: 02/04/10.

[FIP10] Foundations for Intelligent Physical Agents FIPA. Fipa homepage, 2010. `http://www.fipa.org/index.html`; Accessed: 02/04/10.

[FMP96] Klaus Fischer, Jörg P. Müller, and Markus Pischel. Agenda – a general testbed for distributed artificial intelligence applications. In G.M.P. O'Hare and Nick R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, pages 401 – 427. John Wiley & Sons, New York, 1996.

[FMPS95] Klaus Fischer, Jorg P. Muller, Markus Pischel, and Darius Schier. A model for cooperative transportation scheduling. In Victor R. Lesser and Les Gasser, editors, *Proceedings of the First International Conference on Multiagent Systems (ICMAS)*, pages 109–116, San Francisco, California, USA, 1995. The MIT Press.

[FSWZ03] Daniel Frey, Tim Stockheim, Peer-Oliver Woelk, and Roland Zimmermann. Integrated multi-agent-based supply chain management. In *Proc. 5th. Int. Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprise (WET ICE'03)*, pages 24 – 29. IEEE Computer Society Press, 2003.

[FUP91] Roger Fisher, William Ury, and Bruce Patton. *Getting to yes: negotiating, agreement without giving in.* Penguin Books, New York, 2nd. edition, 1991.

[FWJ05] Shaheen S. Fatima, Michael Wooldridge, and Nick R. Jennings. Bargaining with incomplete information. *Annals of Mathematics and Artificial Intelligence*, 44:207 – 232, 2005.

[Gai04] Michael Gaitanides. Prozessorganisation. In Georg Schreyögg and Axel von Werder, editors, *Handwörterbuch Unternehmensführung und Organisation*, pages 1208–1218. Schäffer-Poeschel, Stuttgart, 4th edition, 2004.

[GB01] J. van Gurp and J. Bosch. Design, implementation and evolution of object oriented frameworks: concepts and guidelines. *Software - Practive and Experience*, 31:277–300, 2001.

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of reusable object-oriented software.* Addison-Wesley Professional Computing Series. Addision Wesley Longman Inc., Reading,, 1994.

[GLL00] Giuseppe De Giacomo, Yves Lespérance, and Hector J. Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1-2):109 – 169, 2000.

[GLLK79] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimizing and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287 – 326, 1979.

[GMO03] Paolo Giorgini, Jörg P. Müller, and James Odell, editors. *Agent-Oriented Software Engineering IV, 4th International Workshop, AOSE 2003.* Lecture Notes in Computer Science. Springer, Berlin, 2003.

[GNT04] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated planning : theory and practice.* Elsevier, Kaufmann, Amsterdam et al., 2004.

[Gor99] H. T. Goranson. *The agile virtual enterprise: cases, metrics, tools.* Quorum Books, Westport, London, 1999.

[GOW03] Fausto Giunchiglia, James Odell, and Gerhard Weiss, editors. *Agent-Oriented Software Engineering III, Third International Workshop, AOSE 2002,*, volume 2585 of *Lecture Notes in Computer Science.* Springer, Berlin, 2003.

[GSGW04] Jorge J. Gómez-Sanz, Marie-Pierre Gervais, and Gerhard Weiss. A survey on agent-oriented software engineering research. In Federico Bergenti, Marie-Pierre Gleizes, and Franco Zambonelli, editors, *Methodologies and Software Engineering for agent systems: The Agent-oriented softwre Engineering Handbook*, Multiagent systems, artificial societies, and simulated organizations, pages 33–62. Kluwer Academic Publishers, Bosten, 2004.

[GSVW07] Oleg Gujo, Michael Schwind, Jens Vykoukal, and Oliver Wendt. Mehrrundige Kombinatorische Auktionen beim innerbetrieblichen Austausch von Logistikdienstleistungen. In Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, and Marc Ronthaler, editors, *Informatik 2007*, volume P-109 of *Lecture Notes in Informatics (LNI)*, pages 70 – 74, Bremen, 2007. Gesellschaft für Informatik e.V. (GI).

[GSW$^+$01] Sven Grolik, Tim Stockheim, Oliver Wendt, Sahin Albayrak, and Stefan Fricke. Dispositive Supply-Web-Koordination durch Multiagentensysteme. *Wirtschaftsinformatik*, 43(2):143 – 155, 2001.

[GVO07] Luca Gardelli, Mirko Viroli, and Andrea Omicini. Design patterns for self-organising systems. In Hans-Dieter Burkhard, Gabriela Lindemann, Rineke Verbrugge, and László Z. Varga, editors, *Proceedings of the 5th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2007*, volume 4696 of *Lecture Notes in Artificial Intelligence*, pages 123 – 132, Leipzig, 2007. Springer.

[GW07] Martin Josef Geiger and Wolf Wenger. On the interactive resolution of multi-objective vehicle routing problems. In Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, and Tadahiko Murata, editors, *Evolutionary Multi-Criterion Optimization: 4th International Conference, EMO 2007*, volume 4403 of *Lecture Notes in Computer Science*, pages 687 – 699. Springer Verlag, Berlin, Heidelberg, New York, 2007.

[Hah08] Christian Hahn. A domain specific modeling language for multiagent systems. In Lin Padgham, David C. Parkes, Jörg P. Müller, and Simon Parsons, editors, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 233 – 240, Estoril, Portugal, 2008.

[HCY99] Sandra C. Hayden, Christina Carrick, and Qiang Yang. Architectural design patterns for multiagent coordination. In *Proceedings of the 3rd International Conference on Autonomous Agents, AGENTS'99*, 1999.

[HDP06] Lawrence Henesey, Paul Davidsson, and Jan A. Persson. Agent based simulation architecture for evaluating operational policies in transshipping containers. In Klaus Fischer, Ingo Timm, Elisabeth

André, and Ning Zhong, editors, *Multiagent System Technologies (MATES)*, LNAI, pages 73 – 85, Erfurt, 2006. Springer.

[Hee06] Menno Heeren. *Swarm Intelligence als Strategie zur Lösung reaktiver Planungsprobleme in Wertschöpfungsketten.* Dissertation, Carl von Ossietzki Universität Oldenburg, 2006.

[Hei86] Edmund Heinen. *Einführung in die Betriebswirtschaftslehre.* Gabler, Wiesbaden, 9., rev. ed. edition, 1986.

[Hel09] Malte Helmert. Pddl resources, 2009. `http://ipc.informatik.uni-freiburg.de/PddlResources`; Accessed: 03/03/10.

[Hen98] Herwig Henseler. *Aktive Ablaufplanung mit Multi-Agenten*, volume 180 of *DISKI.* infix Verlag, 1998.

[HI06] Manfred J. Holler and Gerhard Illing. *Einführung in die Spieltheorie.* Springer, Berlin, 6., rev. ed. edition, 2006.

[HM73] Arnoldo C Hax and Harlan C. Meal. Hierarchical integration of production planning and scheduling. Technical report, Massachusetts Institute of Technologie, Operations Research Center, September 1973.

[HN09] Hans Robert Hansen and Gustaf Neumann. *Wirtschaftsinformatik 1: Grundlagen und Anwendungen.* Grundwissen Ökonomik. UTB/Lucius & Lucius, Stuttgart, 10th edition, 2009.

[HO05] Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 13(3):483 –514, 2005.

[HOB04] Marc-Philippe Huget, James Odell, and Bernhard Bauer. The auml approach. In Federico Bergenti, Marie-Pierre Gleizes, and Franco Zambonelli, editors, *Methodologies and Software Engineering for Agent Systems*, Multiagent Systems, Artificial Societies, and simulated organizations. Kleuwer Academic Publshers, Bosten, 2004.

[Hor06] Arne Hormann. *Testbasierte Spezifikation von Agenteninteraktionsverhalten.* Diploma thesis, University Bremen,, 2006.

[HS96] Afsaneh Haddadi and Kurt Sundermeyer. Belief-desire-intention agent architectures. In G.M.P. O'Hare and Nick R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, Sixth-Generation Computer Technology Series, pages 169 – 185. John Wiley & Sons, New York, 1996.

[HS99] Michael N. Huhns and Larry M. Stephens. Multiagent systems and societies of agents. In Gerhard Weiss, editor, *Multiagent Systems: a modern approach to distributed artificial intelligence*, pages 79 – 120. MIT Press, 1999.

[HSKF05] Vincent Hilaire, Olivier Simonin, Abder Koukam, and Jacques Ferber. A formal approach to design and reuse agent and multiagent models. In Michael Luck and James Odell, editors, *Agent-Oriented Software Engineering V; Proc. of the 5th International Workshop, AOSE 2004*, volume 3382, pages 142 – 157, New York, NY, USA,, 2005. Springer.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation.* Addison-Wesley Series in Computer Science. Addison-Wesley Publishing Company, Reading, 1979.

[HZWF10] Christian Hahn, Ingo Zinnikus, Stefan Warwas, and Klaus Fischer. Automatic generation of executable behavior: A protocol-driven approach. In Marie-Pierre Gleizes and Jorge J. Gómez-Sanz, editors, *Proc. of the 10th International Workshop on Agent-oriented Software Engineering (AOSE'09) at AAMAS 2009*, Budapest, 2010. Springer. to appear.

[Jen96] Nick R. Jennings. Coordination techniques for distributed artificial intelligence. In G.M.P. O'Hare and Nick R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, Sixth-Generation Computer Technology Series, pages 187 – 210. John Wiley & Sons, New York, 1996.

[JJ05] Hosang Jung and Bongju Jeong. Decentralised production-distribution planning system using collaborative agents in supply chain network. *International Journal of Manufacturing Technology*, 25:167– 173, 2005.

[JSW98] Nick R. Jennings, K. Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1:7–38, 1998.

[JTY05] Catholijn M. Jonker, Jan Treur, and P?nar Yolum. A formal reuse-based approach for interactively designing organizations. In Michael Luck and James Odell, editors, *Agent-Oriented Software Engineering V; Proc. of the 5th International Workshop, AOSE 2004*, volume 3382, pages 221 – 237, New York, NY, USA,, 2005. Springer.

[KDF05] Manuel Kolp, T. Tung Do, and Stéphane Faulkner. Introspecting agent-oriented design patterns. In S. K. Chang, editor, *Handbook of Software Engineering and Knowledge Engineering,*, volume Vol. 3: Recent Advances, pages 151–176. World Scientific Publishing Co, 2005.

[KF04] Lalana Kagal and Tim Finin. Modeling conversation policies using permissions and obligations. In Rogier M. van Eijk, Marc-Philippe Huget, and Frank Dignum, editors, *AAMAS 2004 Workshop on Agent Communication (AC2004)*, New York, 2004.

[KHLS06] Stefan Kirn, Otthein Herzog, Peter Lockemann, and Otto Spaniol, editors. *Multiagent Engineering.* International Handbooks on Information Systems. Springer, Berlin, 2006.

[Kir06] Stefan Kirn. Flexibility of multiagent systems. In Stefan Kirn, Otthein Herzog, Peter Lockemann, and Otto Spaniol, editors, *Multiagent Engineering*, pages 53 – 69. Springer, Berlin, 2006.

[KKAO00] Takahiro Kawamura, Naoki Kase, Dai Araki, and Akihiko Osuga. Delopment of a distributed cooperative scheduling system based on negatioations between scheduling agents. *Systems and Computers in Japan*, 31(1):92 – 101, 2000.

[Kou08] Anastasios Koutis. *Erstellen von formalen Strukturen und Evaluierung von Abhängigkeiten für gekoppelte Planungsprobleme.* Diploma thesi, Gothe University, 2008.

[Krc10] Helmut Krcmar. *Informationsmanagement.* Springer, Heidelberg, 5th. edition, 2010.

[KRH03] Jean-Luc Koning and Ivan Romero-Hernandez. Generating machine processable representations of textual representations of auml. In Fausto Giunchiglia, James Odell, and Gerhard Weiss, editors, *Agent-Oriented Software Engineering III, Third International Workshop, AOSE 2002,*, volume 2585 of *Lecture Notes in Computer Science*, pages 126–137, Bologna, Italy, 2003. Springer.

[KST07] Hermann Krallmann, Marten Schönherr, and Matthias Trier, editors. *Systemanalyse im Unternehmen: Prozessorientierte Methoden der Wirtschaftsinformatik*. Oldenbourg, München, 2007.

[Kur05] Karl Kurbel. *Produktionsplanung und -steuerung im Enterprise Resource Plannug und Supply Chain Management*. Oldenbourg Verlag, München, Wien, 6. edition, 2005.

[Kur08] Zijad Kurtanovic. *Spezifikation von Verhandlungsstrategien*. Diploma thesis, Gothe University, 2008.

[KYR88] N. P. Keng, D.Y. Yun, and M. Rossi. Interaction sensitive planning system for jop-shop scheduling. In M. D. Oliff, editor, *Expert Systems and Intelligent Manufacturing*. Elsevier, 1988.

[Lam09] Axel von Lamsweerde. *Requirments Engineering: From System Goals to UML Models to Software Specification*. John Wiley & Sons Ldt., Chichester, 2009.

[Lar01] Allan Larsen. *The Dynamic Vehicle Routing Problem*. PhD thesis, Technical University of Denmark (DTU), 2001.

[LDM03] Michael Luck, Mark D'Inverno, and Steve Munroe. Autonomy: Variable and generative. In Henry Hexmoor, Rino Falcone, and Cristiano Castelfranchi, editors, *Agent Autonomy*, Multiagent Systems, Artificial Societies, and Simulated Organizations, pages 9 – 22. Springer, 2003.

[LGS09] Michael Luck and Jorge J. Gómez-Sanz, editors. *Agent-Oriented Software Engineering IX, 9th International Workshop, AOSE 2008*. Lecture Notes in Computer Science. Springer, Berlin, 2009.

[Lin03] Jürgen Lind. Patterns in agent-oriented software engineering. In Fausto Giunchiglia, James Odell, and Gerhard Weiss, editors, *Agent-Oriented Software Engineering III Proc. of the Third International Workshop, AOSE 2002*, volume 2585 of *Lecture Notes in Computer Science*, pages 47 – 58, Bologna, Italy, 2003. Springer.

[LJS06] Xudong Luo, Nicholas R. Jennings, and Nigel Shadbolt. Acquiring user tradeoff strategies and preferences for negotiating agents: A default-then-adjust method. *International Journal of Human Computer Studies*, 64(4):304–321, 2006.

[LP08] Michael Luck and Lin Padgham, editors. *Agent-Oriented Software Engineering VIII, 8th International Workshop, AOSE 2007*. Lecture Notes in Computer Science. Springer, Berlin, 2008.

[LS96] Jyi-Shane Liu and Katia Sycara. Multiagent coordination in tightly coupled task scheduling. In *1996 International Conference on Multi-Agent Systems*, 1996.

[LWD+06] Leif-Erik Lorenzen, Peer-Oliver Woelk, Berend Denkena, Thorsten Scholz, Ingo J. Timm, and Otthein Herzog. Integrated process planning and production control. In Stefan Kirn, Otthein Herzog, Peter Lockemann, and Otto Spaniol, editors, *Multiagent Engineering: Theory and Applications in Enterprises*, International Handbook on Information Systems, pages 91–113. Springer, Heidelberg, 2006.

[Mai04] Roger T. Mailler. *A mediaton-based approach to cooperative, distributed problem solving*. PhD thesis, University of Massachusetts Amherst, 2004.

[Mar92] Frank von Martial. *Coordinating Plans of Autonomous Agents*. Lecture Notes in Artificial Intelligence. Springer, Berlin, 1992.

[MB06] Frank Meisel and Christian Bierwirth. Integration of berth allocation and crane assignment to improve the resource utilization at a seaport container terminal. In Hans-Dietrich Haasis, Herbert Kopfer, and Jörn Schönberger, editors, *Operations Research Proceedings 2005*, volume Volume 2005 of *Operations Research Proceedings*, pages 105–110, Bremen, 2006. Springer.

[MC94] Thomas W. Malone and Kevin Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87 – 119, 1994.

[Mei08] Leif Meier. *Koordination interdependenter Planungssysteme in der Logistk*. Gabler, Wiesbaden, 2008.

[MGH+98] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. Pddl – the planning domain definition language – version 1.2 (pdf). Technical report, Yale Center for Computational Vision and Control, 1998.

[Mül96] H. Jürgen Müller. Negotiation principles. In G.M.P. O'Hare and Nick R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, Sixth-Generation Computer Technologies, pages 211 – 229. John Wiley & Sons, New York, 1996.

[MP01] Massimo Mecella and Barbara Pernici. Designing wrapper components for e-services in integrating heterogeneous systems. *The VLDB Journal - The International Journal on Very Large Data Bases*, 10(1):2 – 15, 2001.

[MPT95] Jörg P. Müller, Markus Pischel, and Michael Thiel. Modeling reactive behaviour in vertically layered agent architectures. In Michael Wooldridge and Nick R. Jennings, editors, *Intelligent Agents I: Agent Theories, Architectures, and Languages*, Lecture Notes in Artificial Intelligence (LNAI), pages 261–276. Springer, Berlin, 1995.

[MS07] Leif Meier and René Schumann. Coordination of interdependent planning systems, a case study. In Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, and Marc Ronthaler, editors, *Informatik 2007*, volume P-109 of *Lecture Notes in Informatics (LNI)*, pages 389 – 396, Bremen, 2007. Gesellschaft für Informatik e.V. (GI).

[MSB$^+$08] Rajiv T. Maheswaran, Pedro Szekely, M. Becker, S. Fitzpatrick, G. Gati, J. Jin, R. Neches, N. Noori, C. Rogers, R. Sanchez, K. Smyth, and C. Vanbuskirk. Predictability & criticality metrics for coordination in complex environments. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 2: SESSION: Agent cooperation table of contents*, pages 647–654, Estoril, Portugal, 2008.

[MSW02] David P. Myatt, Hyun Song Shin, and Chris Wallace. The assessment: Games and coordination. *Oxford Review of Economic Policy*, 18(4):397 – 417, 2002.

[MZ06] Jörg P. Müller and Franco Zambonelli, editors. *Agent-Oriented Software Engineering VI, 6th International Workshop, AOSE 2005.* Lecture Notes in Computer Science. Springer, Berlin, 2006.

[NK95] Bernhard Nebel and Jana Koehler. Plan reuses versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence*, 75(1-2):427–454, 1995.

[NM01] Natalya F. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05, Stanford Knowledge Systems Laboratory, 2001.

[NM04] Klaus Neumann and Martin Morlock. *Operations Research.* Carl Hanser Verlag, München, Wien, 2.nd edition, 2004.

[NRTV07] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani, editors. *Algorithmic Game Theory.* Cambridge University Press, New York, 2007.

[NRW02] Matthias Nickles, Michael Rovatsos, and Gerhard Weiss. A schema for specifying computational autonomy. In *Proceedings of the Third International Workshop "Engineering Societies in the Agents World" (ESAW'2002)*, Madrid, Spain, 2002.

[NT04] David Naso and Biagio Turchiano. A coordination strategy for distributed multi-agent manufacturing systems. *International Journal of Production Research*, 42(12):2497– 2520, 2004.

[Nwa96] Hyacinth S. Nwana. Software agents: An overview. *Knowledge Engineering Review*, 11(3):205–244, 1996.

[Oes09] Bernd Oestereich. *Analyse und Design mit UML 2.3: Objektorientierte Softwareentwicklung.* Oldenbourg Verlag, München, Wien, 9th edition, 2009.

[OGM04] James Odell, Paolo Giorgini, and Jörg P. Müller, editors. *Agent-Oriented Software Engineering V, 5th International Workshop, AOSE 2004.* Lecture Notes in Computer Science. Springer, Berlin, 2004.

[OMG01] Object Management Group OMG. Omg unified modeling language specification, version 1.4, 2001. `http://www.omg.org/cgi-bin/doc?formal/01-09-67`; Accessed: 03/04/10.

[OMG07] Object Management Group OMG. Omg unified modeling language (omg uml), superstructure, version 2.1.2, 2007. `http://www.omg.org/spec/UML/2.1.2/`; Accessed: 03/04/10.

[OO03] Andrea Omicini and Sascha Ossowski. Objective versus subjective coordination in the engineering of agent system. In Matthias Klusch, Sonia Bergamaschi, Pete Edwards, and Paolo Petta, editors, *Intelligent Information Agents*, Lecture Notes in Artificial Intelligence, pages 179 – 202. Springer, Berlin, 2003.

[OOR04] Andrea Omicini, Sascha Ossowski, and Alessandro Ricci. Coordination infrastructures in the engineering of multiagent systemstems. In Federico Bergenti, Marie-Pierre Gleizes, and Franco Zambonelli, editors, *Methodologies and Software Engineering for agent systems: The Agent-oriented softwre Engineering Handbook*, Multiagent systems, artificial societies, and simulated organizations. Kluwer Academic Publishers, Bosten, 2004.

[OPB01] James Odell, H. van Dyke Parunak, and Bernhard Bauer. Representing agent interaction protocols in uml. In Paolo Ciancarini and Michael Wooldridge, editors, *Agent-Oriented Software Engineering*, Lecture Notes in Computer Science, pages 121– 140. Springer, 2001.

[ORV$^+$04] Andrea Omicini, Alessandro Ricci, Mirko Viroli, Cristiano Castelfranchi, and Luca Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, volume Volume 1, pages 286–293, New York, New York, 2004. IEEE Computer Society.

[ORVR04] Andrea Omicini, Alessandro Ricci, Mirko Viroli, and Giovanni Rimassa. Integrating objective & subjective coordination in multiagent systems. In *2004 ACM Symposium on Applied Computing*, pages 449 – 455, Nicosia, Cyprus, 2004.

[Oss98] Sascha Ossowski. *Co-ordination in Artifical Agent Societies.* Lecture Notes in Artifical Intelligence. Springer, Berlin et al., 1998.

[Oss08] Sascha Ossowski. Coordination in multi-agent systems: Towards a technology of agreement. In Ralph Bergmann, Gabriela Lindemann, Stefan Kirn, and Michal Pechoucek, editors, *Sixth German Conference on Multiagent System Technologies (MATES 08)*, Lecture Notes in Artificial Intelligence, pages 2–12, Kaiserslautern, 2008. Springer.

[Par87] H. van Dyke Parunak. Manufacturing experience with the contract net. In Michael N. Huhns, editor, *Distributed Artificial Intelligence*, pages 285–310. Pitman, London, 1987.

[Par96] H. van Dyke Parunak. Visualizing agent conversations: Using enhanced dooley graphs for agent design and analysis. In *Pro-*

ceedings of the Second International Conference on Multi-Agent Systems (ICMAS 96), 1996.

[Pin05] Michael Pinedo. *Planning and Scheduling in Manufacturing and Services.* Springer Series in Operations Research. Springer, New York, 2005.

[PJDH03] Thorsten O. Paulussen, Nick R. Jennings, Keith S. Decker, and Armin Heinzle. Distributed patient scheduling in hospitals. In *18th Int. Joint Conf. on AI (IJCIA 03)*, Acapulco, Mexico, 2003.

[PK03] Young-Man Park and Kap Hwan Kim. A scheduling method for berth and quay cranes. *OR Spectrum*, 25(1):1–23, 2003.

[PKM07] András Pfeiffer, Botond Kádár, and László Monostri. Stability-oriented evaluation of rescheduling strategies, by using simulation. *Computers in Industry*, 58:630–643, 2007.

[Pol96] Martha E. Pollack. Planning in dynamic environments: The dipart system. In A. Tate, editor, *Advanced Planning Technology: Technology Achievements of the ARPA/Rome Laboratory Planning Initiative*, pages 218–225. AAAI Press, 1996.

[PRBH09] Michele Piunti, Alessandro Ricci, Olivier Boissier, and Jomi F. Hubner. Embodied organisations in mas environments. In Lars Braubach, Wiebe van der Hoek, Paolo Petta, and Alexander Pokahr, editors, *Multiagent System Technologies (MATES 2009)*, Lecture Notes in Artificial Intelligence (LNAI), pages 115 – 127, Hamburg, 2009. Springer.

[Pre95] Wolfgang Pree. *Design Patterns for Object-Oriented Software Development.* ACM Press Book. Addison Wesley Longman, Reading, 2nd edition, 1995.

[Pri08] Christine Pries. *Eine Ontologyie für intelligente Ablaufplanung.* Diploma thesis, Carl von Ossietzky Universität, 2008.

[PS10] Carolin Püttmann and Hartmut Stadtler. A collaborative planning approach for intermodel freight transportation. *OR Spectrum*, 32(3):809–830, 2010.

[PSJ98] Simon Parsons, Carles Sierra, and Nick R. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261 – 292, 1998.

[Püt07] Carolin Püttmann. Collaborative planning in intermodal freight transportation. In Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, and Marc Ronthaler, editors, *Informatik 2007*, volume P-109 of *Lecture Notes in Informatics (LNI)*, pages 62 – 65, Bremen, 2007. Gesellschaft für Informatik e.V. (GI).

[PTDL07] Michael P. Papazoglou, Paolo Traverso, Schahram Dustda, and Frank Leymann. Service-oriented computing: State of the art and research challenges. *IEEE Computer*, 40(11):64 – 71, 2007.

[PZ07] Lin Padgham and Franco Zambonelli, editors. *Agent-Oriented Software Engineering VII, 7th International Workshop, AOSE 2006.* Lecture Notes in Computer Science. Springer, Berlin, 2007.

[RG95] Anand S. Rao and Michael P. Georgeff. BDI Agents: From Theory to Practice. Technical Node 56, Australian Artificial Intelligence Institute, April 1995.

[RMB01] William A. Ruh, Francis X. Maginnis, and William J. Brown. *Enterprise application integration.* A Wiley tech brief. John Wiley & Sons Inc., New York, 2001.

[RN03] Stuart J. Russell and Peter Norvig. *Artificial Intelligence A Modern approach.* Prentice Hall Series in Artificial Intelligece. Peason Education, Upper Sadle River, New Jersey, US, 2. edition, 2003.

[RRS03] Amritpal Singh Raheja, K. Rama Bhupal Reddy, and Velusamy Subramaniam. A generic mechanism for repairing job shop schedules. *Innovation in Manufacturing Systems and Technology (IMST)*, 2003.

[RV99] Christian Ruß and Gero Vierke. The matrix auction: A mechanism for the market-based coordination of enterprise networks. Technical Report RR-99-04, German Research Center for Artifical Intelligence (DFKI), 1999.

[RVO06] Alessandro Ricci, Mirko Viroli, and Andrea Omicini. Programming mas with artifacts. In Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah Seghrouchni, editors, *Programming Multi-Agent Systems: Proc. of the Third InternationalWorkshop (ProMAS 2005)*, volume 3862 of *Lecture Notes in Artificial Intelligence*, pages 206–221, Utrecht, Netherlands, 2006. Springer.

[RZ98a] Jeffrey S. Rosenschein and Gilad Zlotkin. Designing conventions for automated negotiation. In Michael N. Huhns and Munindar P. Singh, editors, *Readings in Agents*, pages 353 – 370. Morgan Kaufmann Publishers, 1998.

[RZ98b] Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers.* MIT Press, Cambridge, London, 2.nd edition, 1998.

[San96] Tuomas Sandholm. *Negotiation amomg self-interested computationally limited agents.* PhD thesis, University of Massachusetts, 1996.

[San99] Tuomas Sandholm. Distributed rational decision making. In Gerhard Weiss, editor, *Multiagent Systems: a modern approach to distributed artificial intelligence*, pages 201– 258. MIT Press, 1999.

[Sau93] Jürgen Sauer. *Wissensbasiertes Lösen von Ablaufplanungsproblemen durch explizite Heuristiken.* DISKI 37. Infix Verlag, 1993.

[Sau02] Jürgen Sauer. *Multi-Site Scheduling Hierarchisch koordinierte Ablaufplanung auf mehreren Ebene.* Habilitationsschrift, Carl von Ossietzky Universität Oldenburg, 2002.

[Sau04] Sylvain Sauvage. Design patterns for multiagent systems design. In *Proceedings of the 3rd International Conference on Artificial Intelligence, MICAI'04*, volume 2972 of *Lecture Notes in Artificial Intelligence (LNAI).* Springer, 2004.

[SB97] Stephen F. Smith and Marcel A. Becker. An ontology for constructing scheduling systems. In *Working Notes of 1997 AAAI Symposium on Ontological Engineering.* AAAI Press, 1997.

[SBPM09] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *EMF: Eclipse Modeling Framework.* the eclipse series. Addison-Wesley Longman, Amsterdam, 2nd edition, 2009.

[Sch99] Claudia Schmidt. *Marktliche Koordination in der dezentalen Produktionsplanung Effizenz - Komplexität - Performance.* Galber, Wiesbaden, 1999.

[Sch01] Michael Schumacher. *Objective Coordination in Multi-Agent System Engineering: Design and Implementation.* Lecture Notes in Artificial Intelligence (LNAI). Springer, Berlin, Heidelberg, 2001.

[Sch04] René Schumann. *Ein agentenbasiertes Verfahren zur Koordination von Planungssystemen.* Diploma thesis, Carl von Ossietzky Universität Oldenburg, 2004.

[Sch07] Michael Schwind. *Dynamic Pricing and Automated Resource Allocation for Complex Information Services.* Lecture Notes in Economics and Mathematical Systems. Springer, Berlin, Heidelberg, 2007.

[Sch09] Jan Benedikt Scheckenbach. *Collaborative Planning in Detailed Scheduling.* Phd thesis, Univeristy Hamburg, 2009.

[Sed98] Robert Sedgewick. *Algorithmen*, volume 2. ed. Addison Wesley, 1998.

[SGZ$^+$06] Stephen F. Smith, Anthony Gallagher, Terry Zimmerman, Laura Barbulescu, and Zachary Rubinstein. Multi-agent management of joint schedules. In Edmund H. Durfee and David J. Musliner, editors, *Proceedings of the AAAI Spring Symposium on Distributed Plan and Schedule Management*, pages 128–135. AAAI Press, 2006.

[She06] G. Richard Shell. *Bargaining for advantage : negotiation strategies for reasonable people.* Penguin Books, New York, 2nd edition, 2006.

[Sin98] Munindar P. Singh. Developing formal specifications to coordinate heterogeneous autonomous agents. In *Proceedings of Third International Conference on MultiAgent Systems (ICMAS'98*, 1998.

[SK02] Hartmut Stadtler and Christoph Kilger, editors. *Supply Chain Management and Advanced Planning: concepts, models, software and case studies*, volume 2. Springer, Berlin, 2002.

[SKL$^+$04] Martijn Schut, Michael Kentrop, Mark Leenaarts, Marco Melis, and Ian Miller. Approach: Decentralised rotation planning for container barges. In R. Lopez de Mataras and L. Saitta, editors, *Proceedings of the Sixteenth European Conference on Artificial Intelligence*, pages 755–759. IOS Press, 2004.

[SKT09] René Schumann, Zijad Kurtanovic, and Ingo J. Timm. Specification of strategies for negotiating agents. In *Workshop Agent-based Technologies and applications for enterprise interOPerability at* *the Eighth International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2009)*, Budapest, 2009.

[SL95] Tuomas Sandholm and Victor R. Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In Victor R. Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*, pages 328 – 335. The MIT Press: Cambridge, MA, USA, 1995.

[SL02] Tuomas Sandholm and Victor Lesser. Leveled-commitment contracting: A backtracking instrument for multiagent systems. *AI Magazine*, 23(3):89–100, 2002.

[SLB09] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: algorithmic, game-theoretic, and logic foundations.* Cambridge University Press, New York, 2009.

[SLT08a] René Schumann, Andreas D. Lattner, and Ingo J. Timm. Challenges in coordinating networked enterprises. In Kulwant S. Pawar, Chandra S. Lalwani, and Ruth Banomyong, editors, *Proceedings of the 13. International Symposium on Logistics (ISL 2008)*, pages 133 – 142, Bangkok, 2008. Centre for Concurrent Enterprise Nottingham University.

[SLT08b] René Schumann, Andreas D. Lattner, and Ingo J. Timm. Cost of control for regulated autonomy. In *Presented at the International Workshop on Organised Adaptation in Multi-Agent Systems (OA-MAS'08, Workshop at the AAMAS 2008)*, LNAI, Estoril, 2008. Springer.

[SLT08c] René Schumann, Andreas D. Lattner, and Ingo J. Timm. Regulated autonomy: A case study. In Lars Mönch and Giselher Pankratz, editors, *Intelligente Systeme zur Entscheidungsunterstützung,Teilkonferenz der Multikonferenz Wirtschaftsinformatik*, pages 83 –98, München, 2008. SCS Publishing House.

[Smi94] Stephen F. Smith. Opis: A methodology and architecture for reactive scheduling. In Monte Zweben and Mark S. Fox, editors, *Intelligent Scheduling*, pages 29 – 66. Morgan Kaufmann Publishers, San Francisco, 1994.

[Som06] Ian Sommerville. *Software Engineering.* Addison-Wesley Longman, Amsterdam, 8. edition edition, 2006.

[SP99] John Sauter and H. van Dyke Parunak. Ants in the supply chain. In *Workshop on Agent based Decision Support for Managing the Internet-Enabled Supply Chain, Agents 99,*, Seattle, 1999.

[SPS10] René Schumann, Matthias Postina, and Jürgen Sauer. Bin packing of potted plants for efficient transportation. *Journal of Applied Packaging Research*, 4(1):27 – 52, 2010.

[SR08] Jan Sudeikat and Wolfgang Renz. On the encapsulation and reuse of decentralized coordination mechanisms: A layered architecture and design implications. *Communications of SIWN*, 4:140–146, 2008.

[SS95] M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17 – 29, 1995.

[SS05] René Schumann and Jürgen Sauer. Abaco, coordination of autonomous entities. In Torsten Eymann, Franziska Klügel, Winfred Lamersdorf, Matthias Klusch, and Michael N. Huhns, editors, *Third German Conference on Multiagent System Technologies (MATES)*, LNAI, pages 222 – 228, Koblenz, 2005. Springer.

[SS07] René Schumann and Jürgen Sauer. Implications and consequences of mass customization on manufacturing control. In Thorsten Blecker, Kaspar Edwards, Gerhard Friedrich, and Fabrizio Salvador, editors, *Innovative Processes and Products for Mass Customization. Proceedings of the Joint Conference of the International Mass Customization Meeting 2007 (IMCM07) and the International Conference on Economic, Technical and Organisational Aspects of Product Configuration Systems (PETO07)*, volume 3 of *Series on Business Informatics and Application Systems*, pages 365 – 378, Hamburg, 2007. GITO.

[SS09] René Schumann and Jürgen Sauer. Online planning: Challenges and lessons learned. In Klaus-Dieter Althoff, Kerstin Bach, and Meike Reichle, editors, *Workshop Proceedings KI 2009 32. Annual Conference on Artificial Intelligence: 23. PuK Workshop*, pages 96 – 107, Paderborn, Germany, 2009.

[SSWG02] Tim Stockheim, Michael Schwind, Oliver Wendt, and Sven Grolik. Coordination of supply webs based on dispositive protocols. In *10th European Conference on Information Systems (ECIS)*, pages 1039 – 1053, Gdansk, 2002.

[Sta07] Vladimir Stantechev. Bereitstellung von informationssystemen - auswahl und eigenentwicklung. In Hermann Krallmann, Marten Schönherr, and Matthias Trier, editors, *Systemanalyse im Unternehmen: Prozessorientierte Methoden der Wirtschaftsinformatik*, pages 281–326. Oldenbourg, München, 5th. edition, 2007.

[Sta09] Hartmut Stadtler. A framework for collaborative planning and state-of-the-art. *OR Spectrum*, 31(1):5–30, 2009.

[STT09] René Schumann, Thomas Timmermann, and Ingo J. Timm. Transportation planning in dynamic environments. In Bernhard Fleischmann, Karl-Heinz Borgwardt, Robert Klein, and Axel Tuma, editors, *Operations Research Proceedings 2008*, Operations Research Proceedings, pages 319 – 324, Augsburg, 2009. Springer.

[SV06] Thomas Stahl and Markus Völter. *Model-Driven Software Development: Technology, Engineering, Management.* John Wiley & Sons Ldt., Chichester, 2006.

[SW07] J. Reenze Steenhuisen and Cees Witteveen. Coordinating planning agents for moderately and tightly-coupled tasks. In *International workshop on Coordinating Agents' Plans and Schedules (CAPS), Workshop at AAMAS 07*, Honolulu, Hawaii, 2007.

[SWH10] Harry M. Sneed, Ellen Wolf, and Heidi Heilmann. *Software-Migration in der Praxis.* dpunkt Verlag GmbH, Heidelberg, 2010.

[Tal09] El-Ghazali Talbi. *Metaheuristics: From Design to Implementation.* Wiley Series on Parallel and Distributed Computing. John Wiley & Sons Inc., Hoboken, 2009.

[Tim04] Ingo J. Timm. *Dynamisches Konfliktmanagement als Verhaltenssteuerung Intelligenter Agenten.* DISKI. Akademische Verlagsgesellschaft Aka, Berlin, 2004.

[TLS09] Ingo J. Timm, Andreas D. Lattner, and René Schumann. Reflection and norms: Towards a model for dynamic adaptation for mas. In Guido Boella, Pablo Noriega, Gabriella Pigozzi, and Harko Verhagen, editors, *Normative Multi-Agent Systems*, Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2009. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[TMD$^+$06] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661 – 692, 2006.

[TMD09] Richard N. Taylor, Nenad Medvidovic, and Eric Dashofy. *Software Architecture: Foundations, Theory, and Practice.* John Wiley & Sons, 2009.

[TS08] Thomas Timmermann and René Schumann. An approach to solve the multi depot vehicle routing problem with time windows (md-vrptw) in static and dynamic scenarios. In *Proceedings of the 22. Workshop Planen, Scheduling und Konfigurieren, Entwerfen (PuK 2008)*, Kaiserslautern, 2008.

[TS09] Ingo J. Timm and René Schumann. Performance measurement of multiagent systems: Towards dependable mas. In *Proc. of the 2009 Spring Simulation Multiconference ADS, BIS, MSE, MSEng*, volume 41 of *Simulation Series*, pages 177–184, San Diego, California, USA, 2009. SCS.

[TSH06] Ingo J. Timm, Thorsten Scholz, and Otthein Herzog. Emerging capabilities in intelligent agents for flexible production control. *Advanced Engineering Informatics Journal*, 20(3):247–259, 2006.

[VHL03] Guilherme E. Vieira, Jeffrey W. Herrmann, and Edward Lin. Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of Scheduling*, 6(1):39–62, 2003.

[VM02] Julita Vassileva and Chhaya Mudgal. Negotiation with incomplete and uncertain information: Trading help in a distributed peer help environment. In Simon Parsons, Piotr Gmytrasiewicz, and Michael Wooldridge, editors, *Game-Theoretic and Decision-Theoretic Agents*, Multiagent Systems, artificial societies, and simulated organizations, pages 337–354. Kluwer Academic Publishers, Boston, 2002.

[VSL$^+$10] Ante Vilenica, Jan Sudeikat, Winfried Lamersdorf, Wolfgang Renz, Lars Braubach, and Alexander Pokahr. Coordination in multi-agent systems: A declarative approach using coordination spaces. In Jörg P. Müller and Paolo Petta, editors, *In Proceedings of International Workshop From Agent Theory to Agent Implementation (AT2AI-7)*, Vienna, Austria, 2010.

[VV06] Paul Verstraete and Paul Valckenaers. Towards cooperating planning and manufacturing execution systems. In *INCOM*, 2006.

[VVB$^+$06] Paul Verstraete, Paul Valckenaers, Hendrik Van Brussel, Karuna Hadeli, and Bart Saint Germain. Multi-agent coordination and control testbed for planning and scheduling strategies. In *Fifth international joint conference on Autonomous agents and multiagent systems*, pages 1451 – 1452, Hakodate, Japan, 2006.

[WCDH00] Robert Ian Whitfield, Graham Coats, Alex H. B. Duffy, and Bill Hills. Coordination approaches and systems - part i: A strategic perspective. *Research in Engineering Design*, 12:48 – 60, 2000.

[WCW05] Chun-Chin Wei, Chen-Fu Chien, and Mao-Jiun J. Wang. An ahp-based approach to erp system selection. *International Journal of Production Economics*, 96:47–62, 2005.

[Wee03] Mathijs de Weerdt. *Plan Merging in Multi-Agent Systems.* PhD thesis, Delft University of Technology, 2003.

[Wei99] Gerhardt Weiß, editor. *Multiagent systems: a modern approach to distributed artificial intelligence.* MIT Press, Cambridge, Massachusetts, 1999.

[WF05] Ian H. Witten and Eibe Frank. *Data Mining. Practical Machine Learning Tools and Techniques.* Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 2nd edition, 2005.

[WGP03] Tom Wagner, Valerie Guralnik, and John Phelps. Tæms agents: Enabling dynamic distributed supply chain management. *Journal of Electronic Commerce Research and Applications*, 2003.

[Wöh02] Günter Wöhe. *Einführung in die Allgemeine Betriebswirtschaftslehre.* Verlag Franz Vahlen, München, 21. edition, 2002.

[WH06] Tom De Wolf and Tom Holvoet. Design patterns for decentralised coordination in self-organising emergent systems. In Sven A.

Brueckner, Salima Hassas, Márk Jelasity, and Daniel Yamins, editors, *Engineering Self-Organising Systems: Proceedings of the 4th InternationalWorkshop, ESOA 2006*, volume 4335 of *Lecture Notes in Artificial Intelligence*, pages 28 – 49, Hakodate, Japan, 2006. Springer.

[WHR09] Cees Witteveen, Wiebe van der Hoek, and Nico Roos. Concurrently decomposable constraint systems. In Lars Braubach, Wiebe van der Hoek, Paolo Petta, and Alexander Pokahr, editors, *Multiagent System Technologies (MATES 2009)*, Lecture Notes in Artificial Intelligence (LNAI), pages 153 – 164, Hamburg, 2009. Springer.

[WHS07] Gerhard Wäscher, Heike Haußner, and Holger Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research (EJOR)*, 183:1109 – 1130, 2007.

[WJ95] Michael Wooldridge and Nick R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10:115–152, 1995.

[WJ05] Gerhardt Weiß and Ralf Jakob. *Agentenorientierte Softwareentwicklung*. Springer Verlag,, Berlin, Heidelberg, New York, 2005.

[WJK00] Michael Wooldridge, Nicholas R. Jennings, and David Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

[Woo99] Michael Wooldridge. Intelligent agents. In Gerhard Weiss, editor, *Multiagent Systems: a modern approach to distributed artificial intelligence*, pages 27 – 77. MIT Press, 1999.

[Woo09] Michael Wooldridge. *An Introduction to Multi Agent Systems*. John Wiley & Sons Ltd, West Sussex, 2nd edition, 2009.

[WRZN06] Peer-Oliver Woelk, Holger Rudzio, Roland Zimmermann, and Jens Nimis. Agent.enterprise in a nutshell. In Stefan Kirn, Otthein Herzog, Peter Lockemann, and Otto Spaniol, editors, *Multiagent Engineering*, International Handbooks on Information Systems, pages 73–90. Springer, Berlin, 2006.

[WW06a] Cees Witteveen and Mathijs de Weerdt. Multi-agent planning for non-cooperative agents. In Edmund H. Durfee and David J. Musliner, editors, *Proceedings of the AAAI Spring Symposium* *on Distributed Plan and Schedule Management*, pages 169 –170. AAAI Press, 2006.

[WW06b] Jan Wörner and Heinz Wörn. Benchmarking of multiagent systems in a production planning and control environment. In Stefan Kirn, Otthein Herzog, Peter Lockemann, and Otto Spaniol, editors, *Multiagent Engineering: Theory and Applications in Enterprises*, International Handbook on Information Systems, pages 115 –134. Springer, Berlin et al, 2006.

[WWC02] Michael Wooldridge, Gerhard Weiss, and Paolo Ciancarini, editors. *Agent-Oriented Software Engineering II, Second International Workshop, AOSE 2001*, volume 2222 of *Lecture Notes in Computer Science*. Springer, Berlin, 2002.

[WWWMM01] Michael P. Wellman, William E. Walsh, Peter R. Wurman, and Jefferey K. MacKie-Mason. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35:271–303, 2001.

[Yok01] Makoto Yokoo. *Distributed Constraint Satisfaction: Foundations of cooperation in muti-agent systems*. Springer, Berlin, 2001.

[ZLW$^+$03] Chuqian Zhang, Jiyin Liu, Yat-wah Wan, Katta G Murty, and Richard J Linn. Storage space allocation in container terminals. *Transportation Research Part B*, 37:883–903, 2003.

[ZS99] Stephan Zelewski and Jukka Siedentopf. Ontology-based coordination of planning activities in networks of autonomous production facilities using multi-agent systems. In Stefan Kirn and Mathias Petsch, editors, *Workshop Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien*, volume 2006, pages 77–84, Ilmenau, 1999.

[ZTP05] Olaf Zimmermann, Mark Tomlinson, and Stefan Peuser. *Perspectives on Web Services*. Springer, Berlin, 2nd. edition, 2005.

Curriculum Vitae of René Schumann

Name	René Schumann
Address	Mümlingstr. 1 60599 Frankfurt am Main, Germany
Born	21.07.1978, Delmenhorst Germany
Marital status	married, two children
Education	
2004	Diploma in Computer Science (Diplom Informatiker) Department of Informatics, Carl von Ossietzky University Oldenburg, Germany
1998	Abitur (equiv. A-Level General Certificate of Education) Commercial High School Delmenhorst, Germany
1995	Realschulabschluss (equiv. GCSE) Realschule an der Königsbergerstr, Delmenhorst, Germany
Work Experience	
2007 - 2010	Research scientist, Information Systems and Simulation group Institute for Informatics Goethe University Frankfurt am Main, Germany
2004 - 2007	Research scientist R&D Division Business Information Management OFFIS Institute for Informatics e.V., Oldenburg, Germany

A. Representation of conversation protocols

In this section's appendix we present the representation and transformation of the iterative contract net protocol. In Section 4.3.2 (on page 142) the specification and transformation of the contract net protocol has been discussed. The additional aspect of the iterative contract net is the usage of loops that are necessary to specify the conversation protocol. In the following we discuss how the expressiveness of the specification mechanism can be extended to express those constructs. With this additional construct all other FIPA interaction protocols can be specified, as well.

A.1. Textual Representation of the iterative contract net protocol

In Listing A.1 (on page 304) we present the code that serves as a base's for generating of the graphical notation of the interaction, as well as the generation of the conversation automata. The `loop` keyword in line 7 is also an UML keyword. UML keywords are summarized, e.g., by Oestereich [Oes09, p. 364]. The `opt` keyword is an UML keyword, as well. It represents an block that is optional executed. We use the `opt` with a specific semantic here. By adding a condition to the description of an edge within the loop statement we define the block that has to be executed if the loop has been left, sending a message with the specific performative as a message. Otherwise this block will not be activated.

A.2. Graphical Representation of the iterative contract net

The Listing A.1 can be transformed into the sequence diagram shown in Figure A.1 (on page 305) using the WebSequeneDiagram tool, already used for the generation of the other sequence diagrams.

Listing A.1 Textual description of a iterated contract net interaction protocol [FIP02g]

```
participant Initiator
participant Participant
note left of Initiator: onetomany
note right of Participant: bilateral

Initiator -> Participant : cfp
loop
  alt
   Participant -> Initiator : reject
  else
   Participant -> Initiator : propose
   alt
     Initiator -> Participant : cfp
   else
    Initiator -> Participant : reject-proposal
   else
    Initiator -> Participant : accept-proposal
   end
  end
end
opt accept-proposal
  alt
    Participant -> Initiator : failure
  else
    Participant -> Initiator : inform
  end
end
```

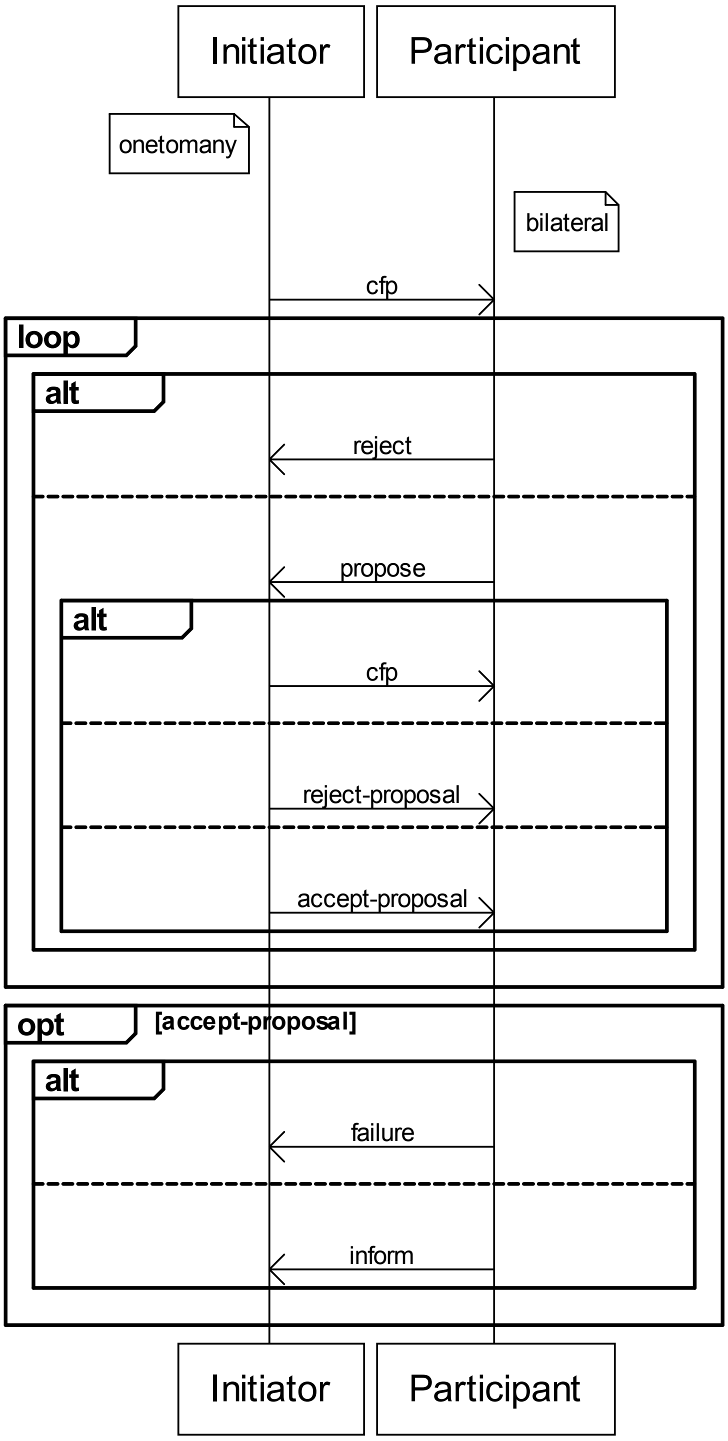


Figure A.1.: Generated sequence diagram of the iterated contract net

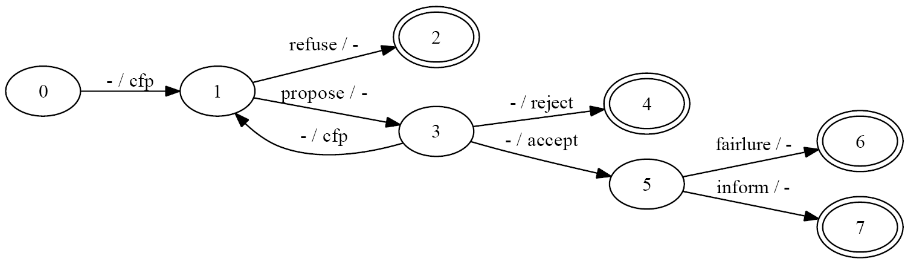


Figure A.2.: Conversation automata for the iterated contract net, initiator role

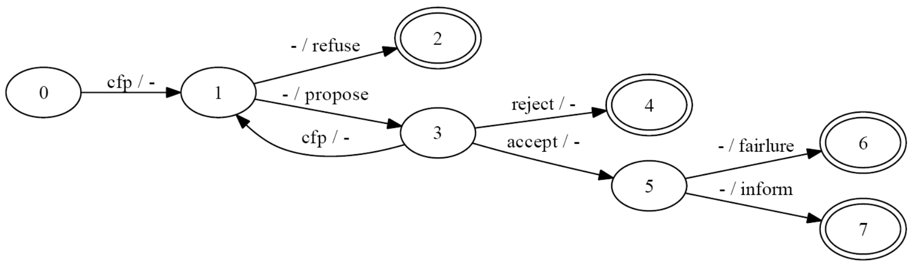


Figure A.3.: Conversation automata for the iterated contract net, participant role

A.3. Representation of the generated automata

As pointed out before, the specification of the conversation protocol is translated within the CoPS framework into conversation automata. A graphical representation of these automata are shown in Figure A.2 and Figure A.3. They are, as they encode a similar protocol, very similar to the automata shown in the Figures 4.7 and 4.8 (on pages 146 and 146).

The only difference is the edge from the node labeled with 3 to the node labeled with 1. This edge leads to a direct circle in the conversation graph, that has been expressed by the loop statement.

B. Evaluation Coordination: The SPT case study

B.1. Task specification

In the following we present the task specification of the practical course "Praktikum Wirtschaftsinformatik und Simulation" [1] at the Goethe University, in the winter term 2009/2010, held by René Schumann and Prof. Dr. Ingo J. Timm.

As the teaching language of this course was German, the description is in German, as well. We give a brief English summary here.

Goods for each order has to be produced, packed and transported. A set of orders is released from the ERP system. For each order the following data is known name/id, product, quantity, customer location, due date for delivery and contract penalty per time unit.

For each product a sequence of steps have to be performed. Each step needs time using a resource. For each product it's volume is known, too. For the given orders a schedule has to be computed. If a goods has been finished it has to be loaded on loading device. A loading device can store a specific volume of goods. Due to limited space only a limited number of loading devices can be packed in parallel. The loading equipment is transported using trucks for the transport to the customers. It is possible to spilt an order into partly deliveries. An order is accomplished if all goods specified in the orders have been delivered to the customer. Transportation costs and duraration are linear and known. Trucks have a limited capacity. They start and end their tour in the depot.

These planning systems have to be accessible as web services. Using agent technology to implement an integrated planning system.

[1] Practical course Business Informatics and Simulation

Handout

Szenario:

Aufträge werden abgearbeitet / produziert und die erstellten Güter anschließend distributiert, d.h. die Waren werden

- Erstellte (Scheduling Problem)
- Verladen (Packproblem)
- Transportiert (TSP)

Eine Menge von Aufträgen wird vom ERP freigegeben. Ein Auftrag umfasst dabei folgende Daten

- Name / ID
- Produkt
- Quantität
- Ziel / Kundenadresse
- Spätester Lieferterim
- Konventionalstrafe per Zeiteinheit

Das ERP kann zu jedem Zeitpunkt weitere Aufträge freigeben. Für jedes Produkt ist die Reihenfolge von Schritten, benötigte Ressourcen und Zeiten sowie dessen Größe als Produktionswissen gegeben. Für die gegebenen Aufträge (Produkt, Quantität) ist ein Plan u erstellen. Ab dem Zeitpunkt der Fertigstellung sind die Waren zu verladen, da keine Bufferfläche zur Verfügung steht. Die Waren werden auf Transporthilfsmittel geladen, die in ausreichender Menge zur Verfügung stehen. Ein Transporthilfsmittel kann zu bis einer fixen Obergrenze beladen werden. DA der Verladeplatz beschränkt ist, können nur eine bestimmte Anzahl an Transporthilfsmitteln parallel beladen werden. Für den Transport stehen eine endliche Zahl von LKWs zur Verfügung. Ist ein Transporthilfsmittel fertig beladen wird es auf einen LKW geladen und der LKW fährt zu den Kunden, die diese Waren geordert haben. Teillieferungen sind möglich. Ein Auftrag ist abgeschlossen, wenn die Waren vollständig ausgeliefert wurden. Die Informationen über die Transportwege liegen als vollständiger Graph vor. Nachdem ein Fahrzeug alle Kunden angefahren hat, kehrt es zum Distributionszentrum zurück und steht für die Durchführung weiterer Transporte zur Verfügung.

Meilensteine

Termin	Meilenstein
21.10.	Vorstellung des Planungsproblem
28.10.	Vorstellung der Konzeption für die Lösung des Planungsprobleme
11.11.	Abnahmetest des Planungssystems
18.11.	Plattformauswahl für WebServices & Dokumentation: Grundlagen WebServices
2.12.	Abnahme: Web Service für das Planungssystems

Anschließend werden die WebServices allen Teilnehmern zur Verfügung gestellt. Dies entspricht einem zweiten Projekt bei dem unterschiedliche Ansätze des Integrated Planning realsiert werden sollen. Hierzu wurden unterschiedliche Beratungsfirmen beauftragt, Vorschläge für die Integration der Planungssysteme zu machen und Prototypisch zu realisieren.

Termin	**Meilenstein**
9.12.	Vorstellung eines Prototyps eines JADE Agenten
16.12.	Vorstellung eines Prototyps für die Nutzung eines Planungs- WebServices mittels eines Agenten
13.1.	Vorstellung eines Prototyps für die Agentenkommunikation auf Basis von Ontologien
20.1.	Vorstellung der Konzeption des Integrated Planning
10.2.	Abnahmetest der Implementierung des Integrated Planning Challange(!)

B.2. Overview of symbols and formulae for the SPT case study

In the following the concepts, sets and function definition used in the SPT case study are summarized.

Table B.1.: Listing of labels

labels	**description**
r	a resource
$o = \langle r, \mathsf{d} \rangle$	an operation
$p = \langle \mathsf{we}, \mathsf{h}, \mathsf{w}, o_1, \ldots, o_n, <_o \rangle$	a product
$c = \langle \mathsf{x}, \mathsf{y} \rangle$	a customer
$d = \langle p, \mathsf{q}, c, \mathsf{d}, \mathsf{e} \rangle$	an order
$g = \langle p, d \rangle$	a good
$a = \langle o, g, r, \mathsf{s}, \mathsf{e} \rangle$	an action
$f = \langle \mathsf{t}, g \rangle$	a finished good
$l = \langle \mathsf{m}, \mathsf{h}, \mathsf{w}, \mathsf{d}, \mathsf{c} \rangle$	a loading device
$pl = \langle l, \mathcal{F}_{pl} \rangle$, with $l \in \mathcal{L}$	packed loading device
$rl = \langle \mathsf{rt}, \mathcal{D}_{rl} \rangle$	ready to ship loading devices
$t = \langle \mathsf{ca}, \mathsf{s}, \mathsf{co} \rangle$	a truck
$tl = \langle \mathsf{t}, \mathcal{RT}_{tl} \rangle$	a truck load
$la = \langle \mathsf{s}, \mathsf{e}, tl, t, tour \rangle$	a loading assignment

Table B.3.: Listing of functions of labels

functions	**description**
$weight(p) = \mathsf{we}$	weight of an product
$quan(d) = \mathsf{q}$	quantity of an order
$customer(d) = c$	customer of order d
$due(d) = \mathsf{d}.$	due of the order
$pen(d) = \mathsf{e}$	penalty of the order
$order(g) = d$	order of a good
$product(g) = p$	product of a good
$orderD(\mathcal{G}, d) = \lvert\mathcal{G}_d\rvert$	number of goods for d

$good(a) = g.$	good of action a
$operation(a) = o$	operation of action a
$start(a) = \mathsf{s}$	start of action a
$end(a) = \mathsf{e}$	end of action a
$resource(a) = r$	resource used for a
$good(f) = g$	good of a finished good
$ttime(f) = \mathsf{t}$	time of finishing
$order(f) = order(good(f)).$	order of f
$cost(l) = \mathsf{c}$	costs of loading device
$items(pl) = \mathcal{F}_{pl}$	payload of l
$mweight(pl) = mw(l)$	maximal weight of payload of pl
$available(pl) = max_{f \in \mathcal{F}pl} ttime(f)$	time of availability of pl
$rtime(rl) = \mathsf{rt}$	earliest shipping time of rl
$customer(rl) = \bigcup_{d \in \mathcal{D}rl} customer(d)$	customers contained in rl
$costLD(rl) = cost(l)$	costs of the device
$orders(rl) = \mathcal{D}_{rl}$	orders contained in rl
$capa(t) = \mathsf{ca}$	capacity of a truck
$speed(t) = \mathsf{s}$	speed of a truck
$costspU(t) = \mathsf{co}$	costs per distance of a truck
$noelements(tl) = \|\mathcal{RT}_{tl}\|$	number of elements of tl
$elements(tl) = \mathcal{RT}_{tl}$	elements of tl
$rtime(tl) = \mathsf{t}$	time when tl can shipped
$customers(tl) = \bigcup_{rt \in \mathcal{RT}tl} customer(rt)$	customers contained in tl
$orders(tl) = \bigcup_{rt \in \mathcal{RT}tl} orders(rt)$	orders contained by tl
$load(la) = tl$	truck load of loading assignment
$truck(la) = t$	truck assigned to la
$start(la) = \mathsf{s}$	start of the loading assignment
$end(la) = \mathsf{e}$	end of the loading assignment
$length(la) = \sum_{i=1}^{i=n-1} : dist(c_i, c_{i+1})$	length of tour
$duration(la) = \mathsf{x}$	duration of a tour
$serviceTime(c, la) = \mathsf{s} + duration'(la)$	service time of a customer
$ft(d, \mathcal{LA}_d) = \max_{la \in \mathcal{LA}_d} serviceTime(c, la)$	finish time of d
$penalty(d, \mathcal{LA}_d) = \mathsf{x}$	penalty of d
$rtdistance = x$	distance of round trip for d

Table B.2.: Listing of used sets

sets	**description**
$\mathcal{R}$	set of resources
$\mathcal{O}$	set of all operations
$\mathcal{P}$	set of products
$\mathcal{C}$	set of customers
$\mathcal{D}$	set of order
$\mathcal{G}$	set of goods
$\mathcal{G}_d$	set of goods for order d
$\mathcal{A}$	set of actions
$\mathcal{A}_g$	partial schedule for g
$\mathcal{F}$	set of finished goods
$\mathcal{L}$	set of types of loading devices
$\mathcal{PL}$	package plan
$\mathcal{RL}$	ready for shipment devices
$\mathcal{T}$	set of trucks
$\mathcal{TL}$	set of truck loads
$\mathcal{LA}$	set of loading assignments
$\mathcal{LA}_t$	plan for a truck
$\mathcal{LA}_d$	lms concerning an element from d

B.3. Data exchange format for the SPT case study

Schnittstellenbeschreibung

Version 0.3
Stand: 04.11.09
Bearbeiter: RS

Datenstrukturen:

Codierung: <Name:Typ>

Auftragsdaten:

```
<Name:String>,<Produkt:String>,<Quantität:Int>,<Ziel-x:Int>,<ziel-
y:Int>,<due-date:Int>,<penalty:Int>
```

Produktdaten:

```
<Name:String>,<Gewicht:Int>,<Hoehe:Int>,<Breite:Int>,<Tiefe:Int>,<op
1:String>,<op2:String>,...<opn:String>
```

Operationsdaten:

```
<Operations-ID:String>,<Resource:String>,<Dauer:Int>
```

Resourcendaten:

Die Verfügbarkeit von Ressourcen ist für jeden Tag gleich. Booleans werden mit 1/0 codiert.

```
<resource-label: String>,<available ze0:Boolean>, .....<available
ze95:Boolean>
```

Ladehilfsmitteldaten

Ladehilfsmittel werden im Folgenden mit Lhm abgekürzt.

```
<Typ:String>,<maxLoad:Int>,<Hoehe:Int>,<Breite:Int>,<Tiefe:Int>,<Kos
ten:Int>
```

Packinformationen

Es gilt das Beladefläche (Fläche für parallel zu bepackende Lhms) + Lagerfläche (Fläche für Warten auf Transport) = Platz für Lhm gesamt

<Platz für Lhm gesamt:Int>,<#beladefläche:Int>,<#lagerfläche:Int>,<kosten pro zwischengelagertem Produkt und Zeiteinheit:Int>

Transporteinheiten (LKW)

In der Entfernungsmatrix ist die Entfernung im Raum angegeben. Die Geschwindigkeit ist notwendig um auf die Zeiteinheiten zu rechnen, die ein Transport benötigt.

```
<id:String>,<kapazität:Int>,<geschwindigkeit:Int>,<costperkm:Int>
```

Transportinformationen(Logistik)

<ServiceZeit:Int>, <Anzahl LKW: Int>

Scheduling2Packen

<Zeit:Int>,<produkt1:String>,<Auftrag:String>

Packen2Tour

<Bereitstellungszeit:Int>,<Auftrag1:String>,…,<Auftragn:String>

B.4. Empirical Evaluation

Inst		mean	std. dev.	# sol.	mode	min	max	p-value
Inst.	seq.	595.71	32.27	10	573	560	660	2.2e-16
1	imp.	529.64	62.53	14	485	418	626	
Inst.	seq.	3111.36	517.37	9	2736	2701	3913	2.2e-16
2	imp.	2124.65	196.68	10	1995	1808	2437	
Inst.	seq.	1074.79	69.54	3	1118	963	1120	2.2e-16
3	imp.	966.93	41.60	2	955	955	1112	
Inst.	seq.	1194.56	119.55	20	1123	986	1609	2.2e-16
4	imp.	863.09	57.96	8	805	805	1014	
Inst.	seq.	2193.43	78.36	8	2226	1998	2409	2.2e-16
5	imp.	1530.56	188.31	17	1558	1249	2284	
Inst.	seq.	684	0.0	1	684	684	684	0.0
6	imp.	630	0.0	1	630	630	630	
Inst.	seq.	990.35	52.93	3	970	970	1156	2.2e-16
7	imp.	901.13	59.35	6	963	832	963	
Inst.	seq.	1908.11	658.31	9	1522	1217	4126	2.2e-16
8	imp.	1356.15	148.31	13	1217	1217	1730	
Inst.	seq.	1507.43	90.21	3	1603	1417	1603	2.2e-16
9	imp.	1367.54	128.66	6	1430	1146	1728	
Inst.	seq.	1511.43	34.07	2	1534	1460	1534	2.2e-16
10	imp.	1115.44	6.14	2	1115	1115	1202	
Inst.	seq.	6374.06	2765.46	19	3931	3929	17787	2.2e-16
11	imp.	3334.91	813.35	9	3698	2237	4356	
Inst.	seq.	1917.79	49.60	3	1959	1834	1959	2.2e-16
12	imp.	1689.60	85.75	36	1746	1338	2046	
Inst.	seq.	19121.31	19469.86	43	10547	3233	64567	2.2e-16
13	imp.	4827.66	4152.75	23	2304	2226	13774	
Inst.	seq.	2246.76	80.14	5	2189	2133	2355	2.2e-16
14	imp.	1743.11	54.85	4	1722	1722	1903	
Inst.	seq.	1808.48	66.38	4	1849	1696	1868	2.2e-16
15	imp.	1647.81	132.90	52	1767	1387	1951	
Inst.	seq.	5042.80	1183.09	2	4167	4167	6641	2.2e-16
16	imp.	2213.45	1033.75	23	1751	1344	3951	
Inst.	seq.	10682.27	2848.29	7	7371	7371	14066	2.2e-16
17	imp.	3325.09	1234.67	10	4618	2037	4618	
Inst.	seq.	1437.20	10.53	10	1444	1355	1445	2.2e-16
18	imp.	1263.67	263.27	80	1101	840	1786	

Inst. 19	seq.	3471.85	464.77	6	3793	2654	3793	2.2e-16
	imp.	2988.75	179.65	35	3041	2416	4336	
Inst. 20	seq.	9087.10	88.70	3	9132	8883	9132	2.2e-16
	imp.	3132.51	1579.83	8	2380	1923	6246	
Inst. 21	seq.	21005.15	1808.57	21	20609	12567	28957	0.9269
	imp.	21139.91	2305.67	20	21784	13467	22900	
Inst. 22	seq.	13583.39	166.9354	5	13487	13318	13806	9.51e-10
	imp.	12533.87	5469.62	67	10110	8810	27510	
Inst. 23	seq.	16309.17	2360.76	4	17562	12491	18071	2.2e-16
	imp.	9756.19	745.83	22	9946	4455	9955	
Inst. 24	seq.	75086.25	6108.25	27	80489	58339	80489	2.2e-16
	imp.	54978.08	4180,80	35	52699	43146	60747	
Inst. 25	seq.	38316.81	22267.95	193	34444	6304	120510	2.2e-16
	imp.	11607.09	9875.49	115	6456	3949	55403	
Inst. 26	seq.	45653.55	6975.75	57	44860	20491	61682	2.2e-16
	imp.	16626.26	5871,65	54	11117	9836	24139	
Inst. 27	seq.	79522.44	6713.10	4	85396	63423	85415	2.2e-16
	imp.	60376.96	10331,11	8	70452	48239	70452	
Inst. 28	seq.	45937	4046.83	18	44504	36589	52698	2.2e-16
	imp.	35449.65	2895.73	3	37209	24733	37209	
Inst. 29	seq.	48467.67	4836.44	4	47945	29157	47945	2.2e-16
	imp.	16272.82	2804.93	9	16153	10670	22258	
Inst. 30	seq.	53174.97	9858.04	57	65760	15773	66480	2.2e-16
	imp.	16875.16	945,01	59	17910	14692	18452	

C. Evaluation Coordination: The SCM case study

C.1. Overview of symbols and formule for the SCM case study

Table C.1.: Listing of labels

labels	description
t	a point in time (*pit*)
a	a PAA
c	a CA
$nu = \langle \mathcal{A}_{n,\mathsf{t}}, c \rangle$	a network
b	a capability
$j = \langle \mathcal{B}_j \rangle$	a job
$v = \langle \mathcal{J}_v, <_v \rangle$	a variant
$p = \langle \mathcal{V}_p \rangle$	a product
$o = \langle \nu, p, a, \mathsf{e}, \mathsf{d} \rangle$	an order
$j^i = \langle o, \mathcal{B}, \mathcal{J}_1, \mathcal{J}_2 \mathsf{e}, \mathsf{d} \rangle$	an instantiated job
$r = \langle j^i, a, \nu, \mathsf{e}, \mathsf{d} \rangle$	a proposal
$m = \langle j^i, a1, a2, \mathsf{e}, \mathsf{d} \rangle$	a commitment

Table C.2.: Listing of used sets

set	description
$\mathcal{A}$	set of PAA
$\mathcal{C}$	set of CA
$\mathcal{N}$	set of networks
$\mathcal{B}$	set of label for capabilities
$\mathcal{B}_a$	capabilities of agent a
$\mathcal{B}_{a,\mathsf{t},\nu}$	capabilities offered by a to ν
$\mathcal{B}_{\nu,\mathsf{t}}$	capabilites of ν

$\mathcal{J}_\nu$	jobs of a network
$\mathcal{V}_\nu$	variants of a network
$\mathcal{J}_{pre,v,j}$	predecessor of j in v
$\mathcal{J}_{suc,v,j}$	successors of j in v
$\mathcal{P}_\nu$	products of ν
$\mathcal{O}$	set of orders
$\mathcal{O}_{\nu,\mathsf{t}}$	orders of network ν
$\mathcal{O}_{a,\mathsf{t}}$	orders of agent a
$\mathcal{J}^i_{a,\mathsf{t}}$	instantiated jobs of a
$\mathcal{J}^i_{\nu,\mathsf{t}}$	instantiated jobs of ν
$\mathcal{A}^C_{j^i,\mathsf{t}}$	candidates of j^i
$\mathcal{R}_{j^i,\mathsf{t}}$	proposals of j^i
$\mathcal{R}_{a,\mathsf{t}}$	proposals of a
$\mathcal{M}$	set of commitments
$\mathcal{M}_{a,\mathsf{t}}$	commitments of a
$\mathcal{M}^{\mathcal{R}}{}_{a,\mathsf{t}}$	commitments of representative a
$\mathcal{M}_{a,\nu,\mathsf{t}}$	commitments of a towards ν
$\mathcal{M}_{\nu,\mathsf{t}}$	commitments of ν
$\mathcal{M}_{o,\mathsf{t}}$	commitments belonging to o

Table C.3.: Listing of functions

function	**description**
$is\_member(a,\nu) = \{true, false\}$	is a member of ν
$n\_c(\nu) = c$	CA of ν
$n\_active(\nu) = \{true, false\}$	is ν active
$b\_has_a(\mathcal{B}_a, \mathsf{t}) = \mathcal{B}_{a,\mathsf{t}}$	capabilities for a
$b\_allow_a(\mathcal{B}_{a,\mathsf{t}}, \nu, \mathsf{t}) = \mathcal{B}_{a,\mathsf{t},\nu}$	capabilities a offers to ν
$j\_b(j) = \mathcal{B}_j$	capabilities required for j
$j\_pre(v,j) = \mathcal{J}_{pre,v,j}$	predecessor of j
$j\_suc(v,j) = \mathcal{J}_{suc,v,j}$	successors of j
$v\_j(v) = \mathcal{J}_v$	jobs of v
$p\_var(p) = \mathcal{V}_p$	variants of p
$o\_\nu(o) = \nu$	network of o
$o\_rep(o) = a$	representative of o
$o\_pro(o) = p$	product of o
$o\_start(o) = \mathsf{e}$	earliest start of o
$o\_end(o) = \mathsf{d}$	due date of o

$o\_active(o) = \{true, false\}$	is order active
$o\_ref(o) = v$	variant for o
$j\_inst(j, o) = j^i$	instantiated job
$j^i\_o(j^i) = o$	order j^i
$j^i\_\nu(j^i) = o\_\nu(j^i\_o(j^i))$	network of j^i
$j^i\_rep(j^i) = o\_rep(j^i\_o(j^i))$	representative of j^i
$j^i\_b(j^i) = \mathcal{B}$	capabilities for j^i
$j^i\_jd(j^i) = \mathcal{J}_1$	jobs j^i depends on
$j^i\_js(j^i) = \mathcal{J}_2$	jobs depend on j^i
$j^i\_start(j^i) = \mathsf{e}$	start of j^i
$j^i\_due(j^i) = \mathsf{d}$	due of j^i
$j^i t\_alc(j^i, a, \mathsf{t}) = \{true, false\}$	can j^i allocated at a
$r\_cand(p) = a$	candidate of r
$r\_j^i(p) = j^i$	instantiated job of r
$r\_start(r) = \mathsf{e}$	start of r
$r\_due(r) = \mathsf{d}$	due of r
$r\_sel(\mathcal{R}_{j^i,\mathsf{t}}, o) = r$	proposal to realize o
$m\_pres(m) = a1$	commitment presenter of m
$m\_rec(m) = a2$	commitment receiver of m
$m\_j^i(m) = j^i$	instantiated job of m
$m\_start(m) = \mathsf{e}$	start of m
$m\_due(m) = \mathsf{d}$	due of m
$m\_active(m) = \{true, false\}$	is commitment active
$m\_gen(r) = m$	generates proposal m
$m\_find(j^i, \mathcal{M}_{\nu,\mathsf{t}}) = m$	finds m for j^i
$m\_comp(\mathcal{M}_{o,\mathsf{t}}, o) = \{true, false\}$	are commitments for o complete
$o\_beg(\mathcal{M}_{o,\mathsf{t}}) = \mathsf{x}$	beginning of o
$o\_end(\mathcal{M}\_o, \mathsf{t}, d) = \mathsf{x}$	end of o
$m\_con(\mathcal{M}_{o,\mathsf{t}}, o)$	is set of commitments consistent
$o\_eval(\mathcal{M}\_o, \mathsf{t}, o) = \mathsf{x}$	evaluates execution of o
$\nu\_eval(\mathcal{M}_{\nu,\mathsf{t}}, \mathcal{O}_{\nu,\mathsf{t}}) = \mathsf{x}$	evaluates performance of ν

Table C.4.: Listing of events

event	**comment**
$\mathfrak{decommit}(m)$	the commitment m is decommited
$\mathfrak{change\_commitment}(m, m')$	request to change m into m'
$\mathfrak{task\_tender}(j^i)$:	a commitment for j^i is searched
$\mathfrak{order\_new}(d)$	a new order d arrives
$\mathfrak{order\_rem}(d)$	remove order d
$\mathfrak{order\_change}(d, d')$	order d is changed to d'.
$\mathfrak{agent\_enter}(n)$	agent a enters network n
$\mathfrak{agent\_leave}(n)$	agent a leaves network n

D. Implementation and evaluation data

The code and its input and resulting output data is included in this thesis in form of a CD. The content of the CD is presented briefly here. On the top level directory of the CD one can find two folders, `Implementation` and `Evaluation`.

D.1. Implementation

In the `Implementation` all code used in the implementation and the CoPS framework is provided. All programs are written in Java. The agent-based development framework JADE is used for the implementation of agents, expect the agents of the ABACO approach, that are implemented as Java threads. The code is attached as an Eclipse project. Eclipse is a widely-used IDE, which has been used here as well. This allows an easier access and structuring of the attached code.

In the `Implementation` folder exist three sub-folders `CoPS`, `Case-Study1`, and `Case-Study2`. In the `CoPS` folder the implementation of the CoPS framework and an additional study for the handling of multiple conversations among agents (in the folder `agent` folder) can be found.

In the folder `Case-Study1` the implementation of the planning systems for scheduling (`SchedulingSource` folder), packing (`packing-source folder`) and transportation (`Tourenplanung-Src-2` folder) can be found. Moreover, the scenario generator (`SPT-Scenario-Generator` folder) and the implementation of the different coordination approaches (`newCoordination` folder), are attached.

In the folder `Case-Study2` the implementation of the ABACO approach (`ABACO` folder) and the competing heuristics, the scenario generator, and the generator for the MPL files (`heuristic` folder) can be found. The ABACO approach needs an MySQL database and additional system libraries that can be found in the ABACO project folder as well. Detailed instructions for the needed localization of the ABACO approach can be found in the `README` file in the project folder.

D.2. Evaluation

In the `Evaluation` folder on the top-level of the CD the input data and resulting outputs used in the evaluation are provided. All data is provided in form of text files. Included are also the *R-scripts*, diagrams, and Excel files used for the statistical evaluation of the data. For the second case study the input and output files of the MPL Modeling System are provided, additionally.

For each case study one sub-folder can be found. For the first case study, presented in Section 5 the input and output data is provided in folder `eval-part1`. Data for the second case study (Section 6) is organized in folder `eval-part2`.