Business Process Fragmentation for Enhancing Process Modeling

Eliane Maalouf
HES-SO Valais Wallis
IIG, Technopôle 3, 3960 Sierre
+41 27 606 90 02
eliane.maalouf@hevs.ch

Marcel Di Zuzio
HEIG-VD
Avenue des Sports 20
CH-1401 Yverdon-les-Bains
dizuzio@infomaniak.ch

Maria Sokhn
HES-SO Valais Wallis
IIG, Technopôle 3, 3960 Sierre
+41 27 606 90 16
maria.sokhn@hevs.ch

ABSTRACT
Business Process Management (BPM) is accepted to be an efficient approach to capture processes in order to improve operational aspects of an enterprise. Business process modeling and design is the first step in BPM. This paper presents a process fragmentation approach that serves to generate process fragments ready for reuse during modeling in a semantic modeling tool. The fragmentation is based on the Refined Process Structure Tree which is an algorithm to decompose processes based on their workflow graphs.

Categories and Subject Descriptors
H.4.1 [**Information System Applications**]: Office Automation---*Workflow management;* J.1 [**Computer Applications**]: Administrative Data Processing---*Government.*

General Terms
Management, Design, Algorithms

Keywords
Process Fragmentation, Process Decomposition, Semantic Web Technologies, Business Process Modeling, Process Auto-completion.

1. INTRODUCTION
Business process modeling is the first phase of the Business Process Management lifecycle and it consists of documenting and designing the process by describing it using, among others, visual elements in computer based graphical tools. Van der Aalst states that in order to support design and management of processes, old processes need to be available for reuse [12]. Additionally, [3] states that the learning curve is still steep for users who are inexperienced in process modeling even if the tools available provide a graphical interface. The authors mention that lack of support during modeling is a contributing factor to this problem. Marcovic et al. [7] defined the requirements to enable reuse of existing knowledge with the following: rich process description, intuitive user request specification, query language with expressive power, query mechanism, flexibility, ranking and computational efficiency. To answer those requirements and overcome the limited support to the users available in existing business process modeling applications, this paper will present the ongoing research on a semantic graphical business modeling tool and more specifically on its process decomposition module. The tool uses: semantic web technologies [8] (RDF, ontologies) to enrich and represent business process elements in the database; SPARQL[1] as the query language to query a database of existing processes for processes that are similar to the user specifications; weights defined by the user for structural, context and historical usage similarities measures; ranking of results from most relevant to least relevant; graphical interface that suggests autocompleting the process while drawing with functionalities of drag and drop of process representations into the canvas. In the background of the auto completion module lays the decomposition module whose goal is to prepare process fragments that can be added to the modeling canvas on the fly. The research challenge is to design and develop a process decomposition approach that allows the extraction of process fragments useful in the context of auto completion during process modeling. In the following sections of this paper are presented the research methodology followed by the design and development of the process decomposition module then the preliminary results to end with a concluding discussion and future works.

Conference'10, Month 1–2, 2010, City, State, Country.

2. STATE OF THE ART
The study of the literature shows that, until the time of writing this paper, there is not a common and widely accepted definition for business process decomposition as outlined by [2,6]. The terms "decomposition" and "fragmentation" are often used interchangeably. Mancioppi etal. [6] defined fragmentation as the "act of creating process fragments out of one process model by applying a fragmentation technique according to some fragmentation criteria." It is this definition that was considered the most relevant for this paper's research work. The fragmentation criteria and the requirements of the decomposition module presented hereafter are based on the classification in [6]:

- **What** input given to the fragmentation: Business processes written in Business Process Model and Notation 2.0 [10] (BPMN 2.0). The choice of this format is imposed by the application domain in which the proof of concept of the tool will be developed. The domain concerns processes of the Swiss E-Government where the specification eCH-0158 [1] recommends the use of BPMN 2.0.

[1]Query language for RDF; http://www.w3.org/TR/rdf-sparql-query/

- **Why** is the process model fragmented: reuse; the processes are fragmented into smaller fragments that can be called by the auto-completion module in the modeling canvas to complete processes on the fly by suggesting the next best fragment to use. The suggestions provided to the user by the modeling tool fall into both categories of subject-based (structural recommendations: complete with a group of elements) and position-based (forward recommendations: complete the next element) in the recommendations classifications in [5]. Figure 1 shows the flow of the auto completion and its relation to the process fragments. P1 is an existing process; P2 is a process being modeled.
- **When** is the fragmentation performed in the process lifecycle**:** modeling phase. In reality the process will be fragmented just after the end of its design in the modeling tool and just before saving it into the database in order to save the whole process along its fragments or when a modeled process is imported into the system.
- **Who** and **How** performs the fragmentation**:** automated software. The goal is to allow for fragmentation to happen without human intervention and integrate with the general workflow of the modeling tool.
- **What** output results from fragmentation**:** the output shall be a set of fragments in BPMN 2.0 independent from each other, without overlapping and sufficiently described to be evaluated in the context of different processes and transformed with semantic web tools.

The study of the existing literature showed that there are two types of decomposition: activity decomposition [4] and workflow decomposition [13, 10, 2]. In the first, fragments are obtained by aggregating activities without considering their succession and the links that exist between them; resulting fragments are not necessarily related to each other. This method is mainly used in process execution context and is not relevant to this research. The second method considers the succession of elements in the process; fragments are made of an aggregation of links and it results in fragments that can be related to each other by a successor/follower relationship or parent/child relationship. This approach is mainly used for process abstraction purposes in analysis settings. Given these fragments properties this approach is adequate with the auto-completion needs since it preserves the relation between fragments. The Refined Process Structured Tree (RPST) [13], part of the second category of approaches, was identified as the most elaborate algorithm to define those fragments. RPST allows the decomposition of a workflow graph into a hierarchy of sub-workflows that are sub graphs with a Single Entry and a Single Exit (SESE) of control. The decomposition is unique and modular and computed in linear time. A workflow graph of a process is a directed acyclic graph where the vertices of the graph are the process activity nodes and the edges of the graph are the transitions between the process nodes [11]. Since RPST decomposes a workflow graph it is independent of the modeling language, it is able to decompose BPMN 2.0 processes once their workflow graphs are generated. RPST decomposes the process into three levels, Polygons (P1, P2 in Figure 2.(a)), Bonds (B1, B2 in Figure 2.(a)) and Triconnected graphs (T1, T2 in Figure 2.(a)). These components are connected to each other through virtual edges (dashed lines in Figure 2.(a)) to form the decomposition tree (Figure 2.(b)). RPST is implemented in an open source software library called JBPT[2].

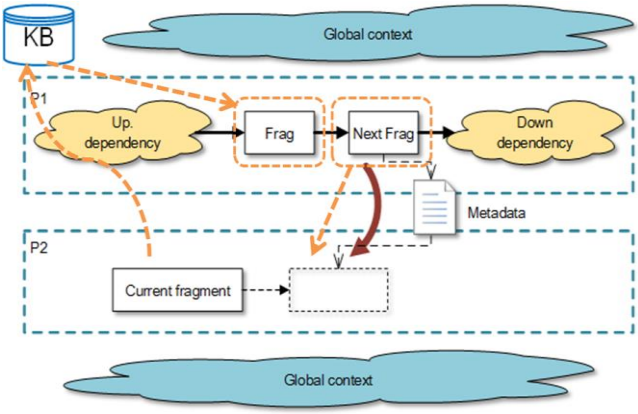

Figure 1. Auto-completion flow in relation to process fragments

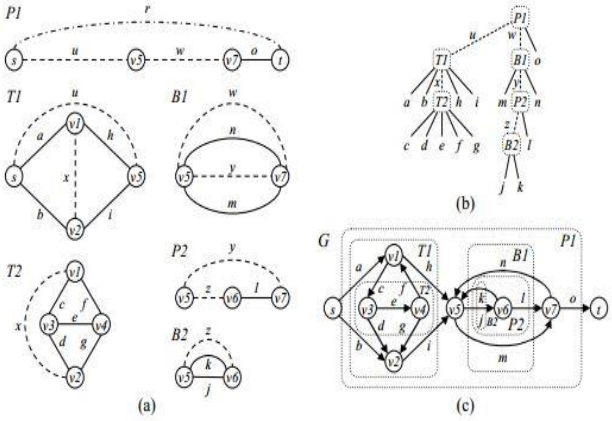

Figure 2. RPST decomposition: (a) Polygons, Bonds and Triconnected graphs; (b) decomposition tree; (c) workflow graph

BPMN 2.0 is a standard [10] to graphically represent the succession of activities of a process (Figure 3). It defines five major categories of graphical elements: flow objects, data objects, connecting objects, swimlanes and artifacts. The process that results from the design is a XML interchangeable file that could be interpreted by diverse modeling tools. The standard introduces an extension mechanism to the constructs of BPMN 2.0. These extensions will be used in this work to insert metadata into the generated fragments.

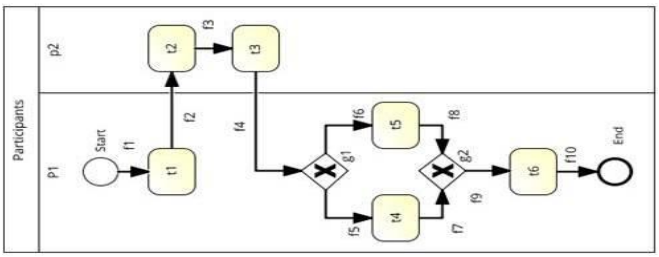

Figure 3. BPMN 2.0 process example

[2] https://code.google.com/p/jbpt/

3. APPROACH DESIGN

The aggregate of requirements for the module are: decompose BPMN 2.0 processes, automatic decomposition, output SESE and related fragments and transform fragments to RDF. The module information flow is represented in Figure 4.

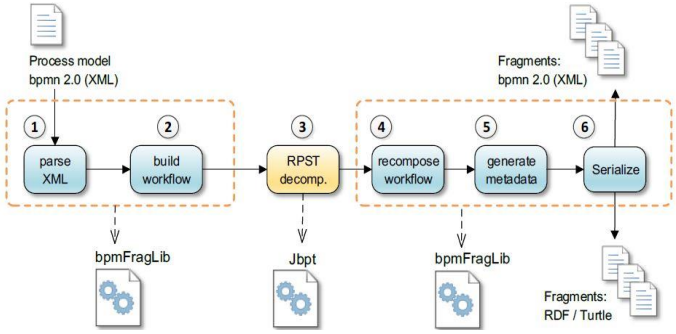

Figure 4. Information flow in the decomposition module

The flow starts by parsing the BPMN 2.0 XML process file (Figure 4.(1)) to create its DOM representation in JAVA objects. In the same step the module collects and stores BPMN elements and their identifiers in order to retrieve them later in the flow. These identifiers and the elements types are then passed to JBPT to construct the workflow graph of the process (Figure 4.(2)). These two steps were implemented in a JAVA library called *bpmnFragLib*. The workflow graph is then fed to the RPST algorithm to generate the decomposition tree (Figure 4.(3)). Since RPST only generates the decomposition tree it cannot be directly used to extract fragments that can be imported into modeling tools and manipulated further in the line since the process elements information that existed in the process file at the beginning is not preserved. In order to overcome this limitation, the decomposition tree is fed to the class *Recompose Workflow* where the fragments are constructed by traveling the decomposition tree and for each of its elements it recollects the necessary information from the original BPMN file using XPATH to rebuild a BPMN 2.0 XML file for each fragment (Figure 4.(4)). From the decomposition tree the class also calculates the predecessor and the following fragments for each fragment and retrieves their identifiers. This information about predecessors and followers is not directly retrieved in the decomposition tree and is reconstructed by the class. The class is able to define multiple predecessors and multiple followers for each fragment given the relation they have in the tree. At this point, fragments are similar to complete BPMN 2.0 XML process; they can be imported in modeling tools and visualized. The decomposition flow proceeds to enrich those fragments with needed metadata to be used by the modeling tool that will manipulate those processes (Figure 4.(5)).

Metadata are added in the form of BPMN Extension elements as mentioned in the section 2.1. The main metadata attributes that were defined are: NextFragments, PreviousFragments, Size (number of flow objects in the fragment), ParentProcRpstDepth (depth of the overall RPST tree gives an idea if the process is flat or includes multiple decisions then more complex), rpstDepthLevel (depth of the fragment seen like a sub tree of the overall process tree), parentProcId (identifier of the parent process that generated the given fragment) and name (a human readable identifier of the fragment). These metadata are shown to the user through a graphical interface and some of them could be edited manually (Figure 5). The final step of the process is the serialization of the fragments and their saving on disk in both BPMN 2.0 XML and RDF format (Figure 4.(6)). The serialization to RDF is a transformation of the XML file into RDF using a XSL transformation file. The XSL was created in a parallel research project and permits the transformation of any BPMN 2.0 into RDF following the BPMN 2.0 ontology [9]. This transformation converts the business process from one single construct to more granular constructs (the process components) that can be queried and addressed individually. The fragments can then be integrated into the triplestore along the original processes and can be queried in the same way through the modeling tool. The XSL file was extended to include the metadata that were added in this project in order to transform them to RDF triples and make them available for querying as well. The classes *Recompose Workflow*, *Generate Metadata* and *Serialize* are also part of the *bpmnFragLib* library.

4. PRELIMINARY RESUTLS

The graphical interface was developed to manage the decomposition process: (Figure 5.(1)) includes a console where are listed all operations done by the module (Figure 5.(2)). On the top is mentioned the name of the BPMN process being decomposed (Figure 5.(3)). It is possible to generate fragments for multiple processes simultaneously in the case of the selection of a repository (Figure 5.(4)). It is possible to choose the method of decomposition (Figure 5.(5)), both methods presented are based on RPST but one shows the full tree and the other pre-processes the full tree and removes some trivial segments like individual process elements and aggregates them with similar contiguous segments to increase their size and their relevance to the auto-completion process. Decomposition is launched by clicking on the "Decompose" button, serialization to XML and RDF is launched by clicking "Serialize Fragments" (Figure 5.(6)). The XML and RDF files are gathered in a repository added to the repository containing the process being decomposed. The global RPST tree is displayed with its depth level and the number of nodes (Figure 5.(7)) and the tree view (Figure 5.(8)). Every time a fragment is selected in the tree, its corresponding meta data are displayed (Figure 5.(9)).

The metadata that can be modified are the name and the score. The score being information that will be updated by the modeling tool and estimates the popularity of the fragment. The rest of metadata (fragment unique id, parent process id, previous fragments, next fragments, fragment depth, process depth, etc.), is generated automatically by the decomposition module.

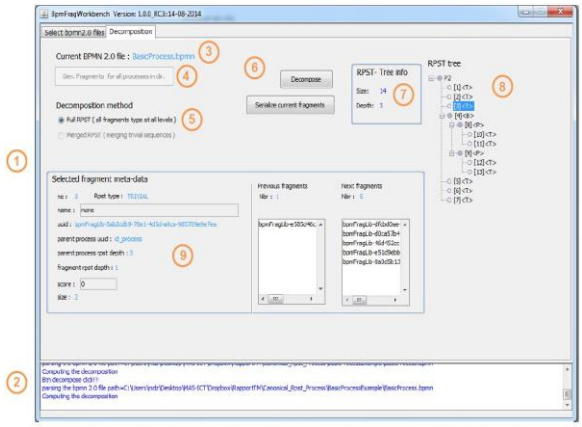

Figure 5. Graphical interface

In total the decomposition module is a set of two JAVA libraries, the *bpmnFragLib* that includes the classes presented earlier and

the *BpmFragWorkbench* library that handles the graphical interface and its interaction with the *bpmnFragLib* and the file system. The process example in Figure 3 was decomposed into the fragments in Figure 6.

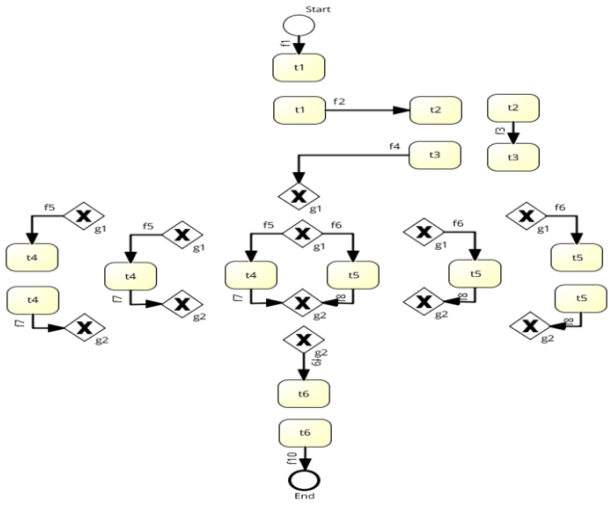

Figure 6. Resulting fragments after decomposition

Figure 6 shows the different granularities that can be retrieved in the process fragments. In an auto-completion scenario this is crucial, since for the completion of a decision node for example (e.g. all elements between g1 and g2 in Figure 6), the user might want to complete with the whole block including all possibilities, or chose one or the other branch or even part of a given branch. Furthermore, the decomposition was tested on real life processes from the domain of E-Government in Switzerland. These processes were larger than the example and some had multiple decision nodes. The first tests showed that the decomposition works correctly with those processes as well.

5. CONCLUSION AND FUTURE WORKS

In this paper we presented the work done until now on a process decomposition module aiming to generate fragments for reuse in a process modeling tool using auto-completion. In the literature, there does not seem to exist a similar work aiming at fragmenting BPMN 2.0 processes in an automated manner and then transforming them to RDF to be used in a semantic environment. In the future, we aim to validate the results of the decomposition on a larger number of processes to identify its limitations and then go further with its integration in the semantic modeling tool being developed to assess the relevance of the generated fragments to the user needs.

6. REFERENCES

[1] Fachgruppe Geschäftsprozesse. (2014). eCH-0158 BPMN-Modellierungskonventionen für die öffentliche Verwaltung. *Online*. (2014). link= http://www.ech.ch/vechweb/page?p=dossier&documentNumber=eCH-0158&documentVersion=1.1

[2] Johannsen, F. and Leist, S. (2012). Wand and Weber's Decomposition Model in the Context of Business Process Modeling. *Business & Information Systems Engineering*, 4(5) (2012), 271-286.

[3] Hornung, T., Koschmider, A. and Lausen, G. (2008). Recommendation Based Process Modeling Support: Method and User Experience. *Conceptual Modeling-ER*.. (2008), 265-278.

[4] Khalaf, R., & Leymann, F. (2006). E role-based decomposition of business processes using bpel. *Web Services*. (2006, September). ICWS'06. International Conference. 770-780. IEEE.

[5] Kluza, K., Baran, M., Bobek, S., & Nalepa, G. J. (2013). Overview of Recommendation Techniques in Business Process Modeling⋆. *Knowledge Engineering and Software Engineering (KESE)*. (2013), 46.

[6] Mancioppi, M., Danylevych, O., Karastoyanova, D. and Leymann, F. (2012). Towards classification criteria for process fragmentation techniques. *Business Process Management Workshop*. (2012, January), 1-12. Springer Berlin Heidelberg.

[7] Markovic, I. and Pereira, A. C. (2008). Towards a formal framework for reuse in business process modeling. *Business Process Management Workshops*. (2008, January), 484-495. Springer Berlin Heidelberg.

[8] Matthews, B. (2005). Semantic web technologies. *E-learning*. 6(6) (2005), 8.

[9] Natschläger-Carpella, C. (2012). Extending BPMN with deontic logic. *Logos-Verlag*.

[10] Object Management Group. (2011). Business Process Model and Notation (BPMN) version 2.0.*Online*. (2011, January) link=http://www.omg.org/spec/BPMN/2.0/

[11] Polyvyanyy, A., Smirnov, S., & Weske, M. (2009). The triconnected abstraction of process models. *Book*. (2009) 229-244. Springer Berlin Heidelberg.

[12] Sadiq, W., & Orlowska, M. E. (2000). Analyzing process models using graph reduction techniques. *Information systems*. 25(2). (2000), 117-134.

[13] Van der Aalst, W.M.P. 2013. Business Process Management: A Comprehensive Survey.*ISRN Software Engineering*. (2013), Article ID 507984.

[14] Vanhatalo, J., Völzer, H., & Koehler, J. (2009). The refined process structure tree. *Data & Knowledge Engineering*. 68(9) (2009) 793-818.