An Agent Based Pervasive Healthcare System: a First Scalability Study*

Johannes Krampf[1], Stefano Bromuri[1], Michael Schumacher[1], and Juan Ruiz[2]

[1] Institute of Business Information Systems,
University of Applied Sciences Western Switzerland
johannes.krampf@students.hevs.ch
{stefano.bromuri,michael.schumacher}@hevs.ch,
WWW home page: http://aislab.hevs.ch/
[2] Department of Endocrinology, Lausanne University Hospital, 1011 Lausanne
juan.ruiz@chuv.ch

Abstract. Gestational Diabetes Mellitus (GDM) occurs during pregnancy due to an increased resistance to insulin caused by the growth of the baby. It appears after the 24th week of pregnancy and it is treated with diet counselling and insulin treatment. In this paper we present the complete implementation of a Pervasive Healthcare System (PHS) based on intelligent agents to support continuous monitoring of pregnant women affected by GDM. Our infrastructure is composed of a mobile interface connecting to a distributed multi-agent system which in turns is connected to a patient management system. This stores the data produced during the monitoring phase and present them to the doctors in charge of the patient. Our system's scalability is then evaluated to show the strong and weak points of our approach.

Key words: Personal Health Systems, Agents, Pervasive Healthcare

1 Introduction

Gestational diabetes mellitus (GDM)[10] affects 3%–10% of all pregnant women with no history of diabetes before pregnancy and manifests itself in high blood sugar levels during pregnancy. Current treatment guidelines [13] consist in diet adjustment and in anti-diabetic medicines such as insulin and metformin. In particular, the patient starts the treatment by simply monitoring the levels of glucose 4 times per day, with one *preprandial* observation and one *postprandial* observation in the morning, and two *postprandial* observations after the lunch and after the dinner. Such values are then written in a notebook that is handed to the doctors twice weekly. According to the behaviour of the physiological values the doctors may introduce further checks at lunch and dinner, and, if the glucose values are outside the boundaries, start the treatment with metformin or insulin. If not treated, GDM may have severe risks for the mother, who may

* This work has been partially funded by the Hasler Stiftung and by the Nano-Tera Consortium.

develop high blood pressure and protenuria (preeclampsia) [14], and for the baby, who may become large for the gestational age (macrosomia), with complications at delivery and later in life.

Rather than checking the patient once or twice weekly, a better monitoring may allow doctors to assess the situation of the patient and propose the correct treatment. One approach to continuous and intelligent patient monitoring is represented by pervasive healthcare [12]. The goal of a pervasive healthcare system (PHS) is to break the boundaries of hospital care, allowing patients to be monitored while living their day-to-day life and to keep in touch with healthcare professionals. Due to its distributed nature, a PHS is faced with three main challenges: scalability, accuracy and security. Scalability is important for PHSs as these systems must be able to serve many patients at the same time, without experiencing disruption due to high loads. Secondly, an accurate PHS should be able to filter information efficiently in order to save time to the healthcare professionals and produce alerts only when needed, with a good trade-off between false positives and false negatives. Finally, security is also an important dimension for a PHS as it deals with medical data, which is sensitive data.

In this paper we present a PHS to monitor patients affected by GDM. A previous version of this system was presented in [2], where we modelled a distributed agent-based PHS. We have chosen agents as a modelling abstraction for our PHS as they are understood to be autonomous software entities, that act proactively and pursue a set of goals [15] in an intelligent way, by applying AI reasoning techniques. Using multi-agent systems (MAS) abstractions to model PHSs is beneficial as this programming paradigm is well suited for distributed systems, due to the autonomy property of the agents, and thanks to distributed event based approach that these systems take into consideration to model the interactions between the agents and the other available resources [3].

In [2] we have already provided a first validation of the accuracy of the notifications provided to the health professionals by our intelligent agents. On one hand, in this paper we present the full implementation of our PHS for GDM, evaluating the scalability of our system and illustrating how healthcare professionals can utilise the functionalities of our tool. On the other hand, the security of our PHS will be evaluated in future publications as the system is currently being audited for security at the Lausanne University Hospital, although in this paper we also present how we secured the interfaces of our PHS. The remainder of this paper is structured as follows: Section 2 discusses the functionalities of the components of our system; Section 3 discusses an evaluation of our PHS in terms of its scalability; Section 4 puts our work in comparison with relevant related works; finally Section 5 concludes this paper and draws the lines for future work.

2 A Personal Health System for GDM

Fig. 1 shows that our system is composed of three main components, which are the *Mobile Infrastructure* (MI), the *Agent Environment* (AE) and the *Patient*

Management System (PMS). Furthermore, these components are interfaced between each others by means of a mediator component, realised as a Web service Data Gateway connector that accepts HTTPS requests. The MI component collects the physiological data of the patient and delivers such data to the AE component and to the PMS component.

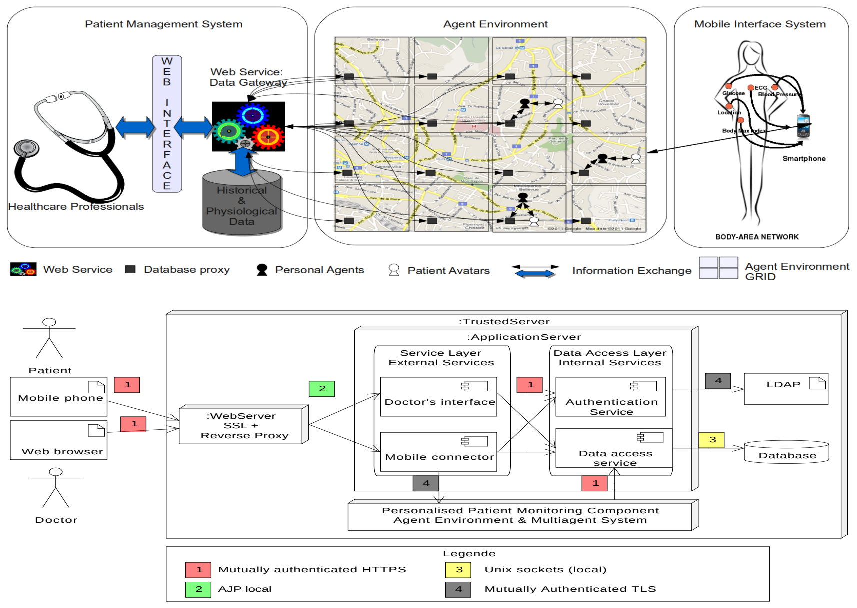


Fig. 1. The Pervasive Health System Logic Architecture and the Security Interfaces.

The AE component utilises logic programming to model intelligent agents that filter the data submitted to the PMS and provide alerts in case of significant events, such as a possibility of preeclampsia in the patient or a high level of blood sugar that requires a treatment adjustment. The AE system is subdivided in cells associated to an area of a real city where the patients connect with their mobile phones to produce their physiological data, that are then evaluated by intelligent agents. The patients are represented in the AE as *avatars* that can communicate to a personal intelligent agent, embodied in the AE. This representation of the patient is convenient as we can reuse the AE communication and notification facilities to interact with the intelligent agents situated in it. To every patient avatar we associate an intelligent agent whose cognitive architecture will be explained later in this Section. Finally the PMS allows the doctors to visualise the patient's data, to modify its treatment and to visualise the alerts produced by the AE.

The three tier logic architecture shown on the top of Fig. 1 translate then to a four tier architecture as shown on the bottom of Fig. 1. In particular, the mobile phone and the Web browser represent the *presentation layer*, the reverse proxy and the Web server represent the *Web application layer*, the agent environment represents the *business logic layer* while the database represents the *data layer*.

The Web application layer accepts outside secure connections only on the HTTPS port. It connects business logic and data layers. Caretakers and patients use client authenticated HTTPS to connect to the system. A second authentication factor is provided by the combination of user name and password. Internal components use local in-memory or mutually authenticated TLS connections to communicate with each other. The data base partition is encrypted to protect against physical access to the hard disk. User access to resources is restricted by membership in one of the three groups users, caretakers and administrators. Access to patient data is further restricted by an access control list which only allows caretakers who treat a patient to access this patient's data. All actions are logged including IP address, user name, resource and success of the action to provide an audit trail.

2.1 The Mobile Infrastructure

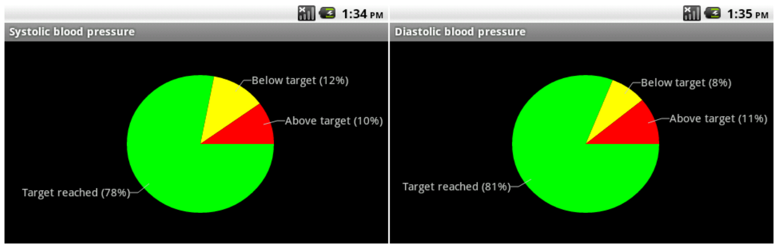
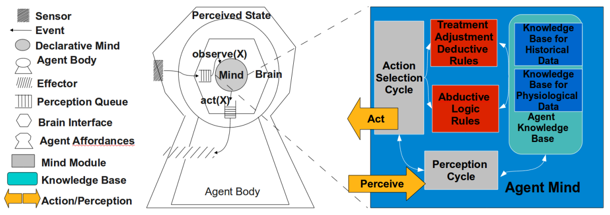


Fig. 2. The Mobile Interface.

The MI component is based on Android, and it is provided to the patient through mobile phones and tablets to introduce their physiologic data and symptoms associated to GDM: blood pressure, blood glucose, weight, pulse, blurred vision, epigastric pain, oedema in the legs, dyspnoea, chest pain, head ache. Such an interface allows the patients to see if their physiological values meet the targets for the week, with a set of pie charts as shown in Fig. 2. At the same time the MI is also built with a synchronisation approach to avoid data loss: whenever the connection with the AE is impossible, the MI saves the data in a local database. When the connection with the AE is possible again, the data stored in the mobile phone is submitted for storage in the PMS.

2.2 The Agent Environment and The Patient Management System

Our PHS makes use of intelligent agents to analyse and filter the physiological data produced by the patients. In particular, we decided to include the GOLEM agent platform in our system as it currently implements the patterns of Distributed Event-Based Systems (DEBSs) as described in [2]. By means of these patterns, the agents in the GOLEM agent platform can subscribe to events produced by the patients and the GOLEM platform will take care of notifying such events when they take place.



Abductive Rules

```
DomainKnowledge :
previous_macrosomia ← macrosomia.
week_of_gestation(W)← W >= 34, macrosomia; W >= 20, preeclampsia.
glucose_observation(A = [Gl1,Gl2, . . . Gl20]) ←
      ∃ Glk,Glj ∈ A, j ≠ i, Glk >5, Glj >5, macrosomia.
previous_preeclampsia; blurred_vision; severe_headache;
      lower_leg_oedema← preeclampsia; severe_preeclampsia.
kidney_disease; oliguria; protenuria; nausea; epigastric_pain ←
      preeclampsia; severe_preeclampsia.
blood_pressure(Systolic, Diastolic)←
      preeclampsia,protenuria, Systolic >= 140, Diastolic >= 90;
      not protenuria,severe_hypertension,Systolic>=160,Diastolic>=110;
      Systolic>=140,Diastolic>=90,protenuria,week_of_gestation(W),W < 35,
      (severe_headache;blurred_vision;nausea;vomiting),severe_preeclampsia.
IC :
      ← preeclampsia, (not protenuria;week_of_gestation(W), W < 20).
      ← protenuria, severe_hypertension.
      ← preeclampsia, blood_pressure(S,D),S < 140, D < 90.

full_piers(ID,Probability,T)←holds_at(ID,platelet,PLA,T),holds_at(ID,spo2,SPO2,T),
   holds_at(ID,creatinine, CRE, T),holds_at(ID,aspartate, AST,T),
   holds_at(ID,gestational_age, Age,T),holds_at(ID,chestpain, Pain,T),
   Coeff = 2.68 + (-5.41*10^-2*Age) + 1.23*Pain + (-2.71*10^-2*CRE) +
   (2.07*10^-1*Plat) + (4*10^-5*Plat^2) + (1.01*10^-2*AST) +
   (-3.05*10^-6*AST^2) + (2.50*10^-4*CRE*PLA)+ (-6.99*10^-5*PLA*AST) +
   (-2.56*10^-3*PLA*SPO2), Probability is exp(Coeff)/1+exp(Coeff).
```

Deductive Rules

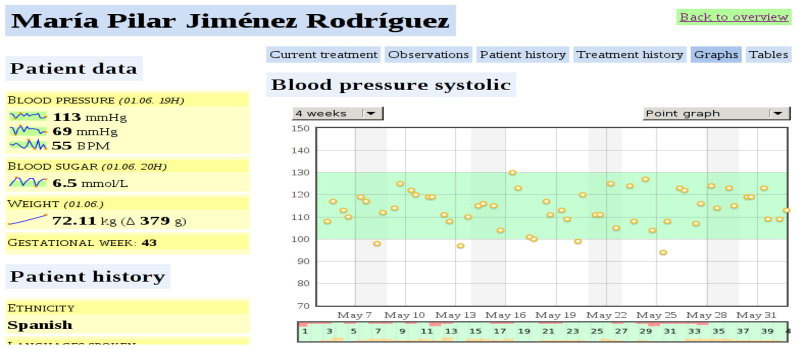
```
R1) terminates(glucose_observation:Ev, ID, observations, _).
R2) initiates(glucose_observation:Ev,ID,observations,[Ev|Tail])←
      patient_of(Ev,ID), time(Ev,T), holds_at(ID,observations, Tail, T).
R3) initiates(glucose_observation:Ev,ID,hypoglycemia_risk, true)←
      patient_of(Ev, ID), time(Ev,T), holds_at(ID,observations, List, T),
      size(List, 20),not (member(M, List), glycemia(M,Gl), Gl >4).
R4) initiates(glucose_observation:Ev, ID, hypoglycemia_risk, true)←
      patient_of(Ev, ID),glycemia(Ev, Gl),time(Ev,T),
      holds_at(ID,observations, [Head|Tail], T), time(Head,T*),
      T - T* < 1, glycemia(Head, Gl2), Gl < 3, Gl2 < 3.
R5) initiates(glucose_observation:Ev,ID,postprandial_observation,[Ev|Tail])←
      is_postprandial(Ev),time(Ev,T),patient_of(Ev, ID),
      holds_at(ID,postprandial_observation, Tail, T).
R6) initiates(glucose_observation:Ev,ID,introduce_preprandial,true)←
      is_postprandial(Ev),time(Ev,T),patient_of(Ev,ID),
      holds_at(ID,postprandial_observation, List, T),
      sublist(Sub,List), size(Sub,6),foreach(Obs,Sub,(glycemia(Obs,G),G>8)).
select(_,A,T)←instance_of(P, patient,T),holds_at(P,introduce_slow_insuline,true,T),
   myID(ID),A=store_event:a1[actor⇒ID,recommendation:p1[
                              timestamp ⇒ T, value ⇒ introduce_slow_insulin]].
```

Fig. 3. Caretaker Agent Mind Architecture.

In the particular case of our PHS, the patients are represented in the AE as avatars that can communicate with a personal caretaker agent, whose architecture is reported in Fig. 3. Every agent is deployed in GOLEM in a container. A GOLEM container represents a portion of the distributed agent environment which in this case is associated with a portion of the real environment, in order to distribute the load of the requests of the patients. This topology was chosen because we imagine that this system could work in synergy with the actual cellular network. As described in [2], every caretaker agent has a cognitive model with a deductive and an abductive part, whose specification is shown in Fig. 3.

The deductive rules are specified in Event Calculus [8], to describe the evolution in time of the patient physiological values. Such rules specify how the treatment of the patient should evolve. For example rule R6 specifies that, when the patient had high glucose in the postprandial observations, then the agent suggests to introduce further preprandial observations. Similarly, if the patient is already in a 6 checks per day regime, then the agent suggests the doctors to introduce a slow insulin in the morning to tackle the values that are out range. The abductive rules take into consideration the symptoms of the patient to provide alerts of macrosomia or preeclampsia to the doctors. In particular, for preeclampsia, we also provide the probability of adverse outcome using the fullPiers model [14], also reported in Fig. 3. To be able to provide this probability, the agent connects to the PMS using the GOLEM middleware to download the blood samples needed by the fullPiers model and introduced in the system through the PMS. Further details about the agent cognitive model and its accuracy are reported in [2] and we refer the interested reader to this publication.

The Patient Management System allows healthcare professionals to visualise and analyse data as well as to introduce new data gathered during a patient's visit. The PMS is a hybrid application incorporating both elements of a classic server side Web application and a modern AJAX-powered client side Web application.


María Pilar Jiménez Rodríguez

Patient data

BLOOD PRESSURE *(01.06. 19H)*
113 mmHg
69 mmHg
55 BPM

BLOOD SUGAR *(01.06. 20H)*
6.5 mmol/L

WEIGHT *(01.06.)*
72.11 kg (Δ **379** g)

GESTATIONAL WEEK: **43**

Patient history

ETHNICITY
Spanish

Blood pressure systolic

4 weeks

Point graph

Fig. 4. The Patient Overview Page.

After logging in to the PMS, healthcare professionals can visualise a dynamic patient page, shown in Fig. 4. The patient page is divided into a static side bar on the left and a changing content area on the right side of the window. When showing different information about the patient, an HTML fragment will be loaded and replace the content area. Graphs are handled completely on the client side. There is a maximum of about 1700 data points, assuming 6 daily blood sugar measurements during 40 gestational weeks, for a graph. Processing and displaying this number of data points is almost instant on modern browsers and allows for better interactivity than server side processing would. Healthcare professionals can interact with the graphs by showing a specific time period, changing this period and changing the view on the data. There exist two further views on the data beside the point view in Fig. 4, which are a percentage view, showing the data points in a week which are below, inside and above the normal range of a physiological value, and and a distribution view that shows the 9th, 25th, 50th, 75th and 91th percentile for each week as well as outliers. These views were created to fulfil the different information needs of doctors at the Lausanne University Hospital.

3 Evaluation

To evaluate our solution, we measured the performance for HTTPS requests with different requirements on the application and database. Our goal in evaluating our PHS was to understand if the system could support the traffic load of the patients of an hospital of a medium sized city, such as the city of Lausanne in Switzerland, where we plan to perform field tests. Also, another goal of this evaluation is to understand what is the maximum amount of patients that we can serve before having to introduce load balancing techniques in the PHS. We therefore perform our evaluation on those components representing a bottleneck of the current architecture. We do not perform an evaluation on the agent environment as this is based on the GOLEM platform, whose performances have been previously evaluated in [3], showing that the system can scale up with the number of GOLEM containers spawned for the application. For the tests, we ran our PHS on a 3 GHz Intel Core 2 Duo processor, 4 GB RAM and Ubuntu 10.04.

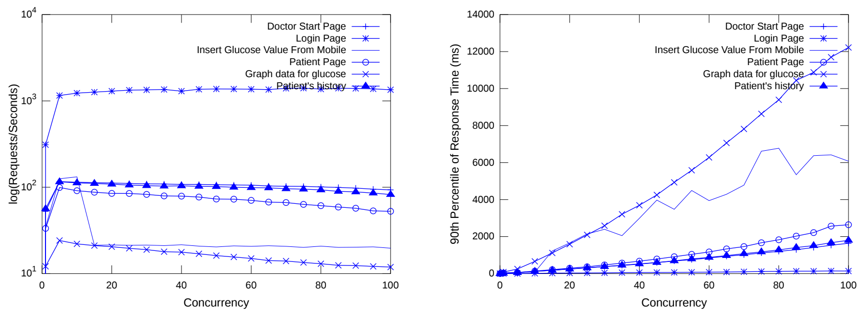


Fig. 5. Requests/second results and 90th percentile response time results.

The tests were performed using the ApacheBench [1] utility, which works by performing a defined number of requests to a specified URL and measuring various values such as response time or transferred bytes for each request. After finishing the benchmark, ApacheBench shows a statistical analysis of all requests, showing transfer rate and mean time per request. To prevent the benchmark utility from influencing the results, it was executed on a notebook with a 1.6 GHz Intel Core 2 Duo processor, 1 GB RAM running Linux with a direct cable connection to the server, to minimise influences caused by network latency. We tested different usage scenarios stressing different parts of the system. All requests were executed at 1 to 100 concurrent requests to simulate different usage load. Values were recorded during a 60 seconds stress testing period.

The curves on the left of Fig. 5 show the response time to different concurrent calls performed in the system. In particular, we show the time to: retrieve the data for glucose in the PMS; retrieve a patient history; insert glucose values from the mobile phone; retrieve a patient summary page; access the login page and patient page. We also assume that concurrent calls coming from different patients are distributed in different cells of the agent environment. The requests per second values increase with higher concurrency levels until a plateau is reached. At this point the server becomes overloaded and the response time increases. The graph of requests to the login page shows the plateauing behaviour when reaching about 1400 requests per second. By looking at the detailed values we discovered that the response time increase between 400% and 500% when comparing 20 and 100 concurrent connections while the processed requests per second are virtually unchanged. The curves on top of Fig. 5 also show the plateauing behaviour of the requests with database activity. The maximum value for requests per second is reached between 10 and 20 concurrent requests. This suggests a database related limit in concurrency. When increasing the concurrency from 10 to 15, the glucose insertion from the mobile phones experiences a sharp fall from 130 to 20 requests per second. The message queue in the Web service data gateway interface is the limiting factor here: messages are acknowledged in order to ensure message delivery and with many concurrent requests messages cannot be acknowledged fast enough.

The 90th percentile response time charts on the right of Fig. 5 show the expected maximum response time for 90% respectively of all requests. The response time begins to be above 1 second and noticeable by a user at about 50 concurrent requests for requests with medium database activity and at about 15 concurrent requests for database heavy requests.

Each patient will transmit a number of values each day: Twice daily blood pressure and four to six times blood sugar. Furthermore, the patient will transmit one weight value per week and she will report symptoms when she experiences them. We are interested in the maximum number of requests in a short time period and will make the pessimistic assumptions that during one second of usage of her mobile phone the patient transmits symptoms, blood pressure, blood sugar and weight at the same time in the morning. The maximum number of requests/second for a user is therefore 4 requests/second. The worst case scenario is all 10 patients of a planned pilot study making their 4 requests concurrently in the same second, leading to 40 requests/second with 10 concurrent connections. When producing Fig. 5 we found that the system is capable of 132 requests/second for a concurrency level of 10. As at the Lausanne University Hospital, that serves the Canton Vaud in Switzerland, there are a maximum of 5–6 patients with GDM at the same time, the system we defined is viable to deal with the load experienced by a big sized university hospital. To estimate the maximum number of users the system can serve, we modify our assumptions to assume an uniform distribution of the 4 requests over the course of 30 minutes. We will furthermore use 20 requests/second as the system's performance due to performance drop off at higher concurrency levels. This results in about 0.002 requests/second per user (Eq. 1) and 9000 patients (Eq. 2) with 20 concurrent connections (Eq. 3). This allows us to consider usage for the whole canton Vaud. The canton has a population of 700,000 [11] and 9.4 births per 1000 inhabitants per year. This results in 6580 births per year (Eq. 4) which means that the system can theoretically monitor all pregnant women in the Canton of Vaud.

$$\frac{4}{30*60}\frac{request \times user}{second} = \frac{1}{450}\frac{request \times user}{second} \tag{1}$$

$$20\frac{request}{second} / \frac{1}{450}\frac{request \times user}{second} = 9000\,user \tag{2}$$

$$(4\frac{request}{user} \times 9000\,user)/30 \times 60\,second = 20\frac{request}{second} \tag{3}$$

$$700000\,inhabitant \times \frac{9.4}{1000}\frac{birth}{inhabitant \times year} = 6580\frac{birth}{year} \tag{4}$$

4 Related Work

From the related work stand point, several attempts have been done in the past to combine agent technology with the healthcare domain. The systems described by Huang et al. in [7] and by Hammond and Sergot in [6] use symbolic reasoning over clinical workflows to manage oncological patients within a healthcare institution and to simplify the management of clinical trials. Larson et al present Guardian in [9], an early attempt to provide an agent-based system for medical monitoring and diagnosis. Guardian uses a tuple space based approach where cognitive agents with a properly programmed knowledge base, provide a diagnosis for situations such as liver failure and hypothermia. In [4] Ciampolini et al present a distributed MAS to deal with distributed diagnosis performed by heterogeneous distributed abductive agents. In Ciampolini's approach the diagnosis is provided in term of probabilities, although they do not consider a realistic model for their experiments. The ASPIC project [5] has developed an architecture based on argumentation theory for an autonomous agent that single- and multi-agent healthcare applications can use. Evaluation scenarios focus on the management and treatment of people with heart disease. With respect to the systems reported above, our contribution is twofold: first of all we developed a practical system that takes into consideration scalability and security issues following the needs of medical doctors at Lausanne University Hospital; secondly, for our intelligent reasoning agents, we also utilise clinical models like the fullPiers [14], whereas the systems mentioned above lack this approach.

5 Conclusion and Future Works

In this paper we presented a fully implemented Personal Health System for the monitoring of GDM and alerting of treatment adjustment suggestions and continuous diagnosis of conditions related to GDM. This system is composed of a mobile infrastructure used by patients, a distributed agent environment and a patient management system for medical professionals in charge of the patients. We evaluated the performances in terms of scalability of our system demonstrating the feasibility of the approach in the case of GDM. Future work implies the evaluation of security interfaces amongst the different layers of our infrastructure and the produced data. Another approach to future work is to have a different PHS to doctors notification system, where the notifications are not just produced in the Patient Management System, but also imply the submission of emails or SMS to the medical professionals.

References

1. ApacheBench. `http://httpd.apache.org/docs/2.2/programs/ab.html`.
2. Stefano Bromuri, Michael Schumacher, Kostas Stathis, and Juan Ruiz. Monitoring gestational diabetes mellitus with cognitive agents and agent environments. In *Proceedings of the 2011th IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2011)*, August 2011.
3. Stefano Bromuri and Kostas Stathis. Distributed Agent Environments in the Ambient Event Calculus. In *DEBS '09: Proceedings of the third international conference on Distributed event-based systems*, New York, NY, USA, 2009. ACM.
4. A. Ciampolini, P. Mello, and S. Storari. Distributed medical diagnosis with abductive logic agents. In *ECAI2002 workshop on Agents in Healthcare, Lione*, 2002.
5. John Fox, David Glasspool, and Sanjay Modgil. A canonical agent model for healthcare applications. *IEEE Intelligent Systems*, 21:21–28, November 2006.
6. Peter Hammond and Marek Sergot. Computer support for protocol-based treatment of cancer. *The Journal of Logic Programming*, 26(2):93 – 111, 1996.
7. Jun Huang, Nicholas R. Jennings, and John Fox. Agent-based approach to health care management. *Applied Artificial Intelligence*, 9(4):401–420, 1995.
8. R Kowalski and M Sergot. A logic-based calculus of events. *New Gen. Comput.*, 4(1):67–95, 1986.
9. Jan Eric Larsson and Barbara Hayes-Roth. Guardian: An intelligent autonomous agent for medical monitoring and diagnosis. *IEEE Intelligent Systems*, 13:58–64, January 1998.
10. D. C. Serlin and R. W. Lash. Diagnosis and management of gestational diabetes mellitus. *Am Fam Physician*, 80:57–62, Jul 2009.
11. Swiss Statistics. The population of Switzerland 2009. `http://www.bfs.admin.ch/bfs/portal/en/index/themen/01/22/publ.Document.136821.pdf`, 2010.
12. Upkar Varshney. *Pervasive Healthcare Computing: EMR/EHR, Wireless and Health Monitoring.* Springer Publishing Company, Incorporated, 2009.
13. N. Vogel, B. Burnand, Y. Vial, J. Ruiz, F. Paccaud, and P. Hohlfeld. Screening for gestational diabetes: variation in guidelines. *Eur. J. Obstet. Gynecol. Reprod. Biol.*, 91:29–36, Jul 2000.
14. P. von Dadelszen et al. Prediction of adverse maternal outcomes in pre-eclampsia: development and validation of the fullPIERS model. *Lancet*, 377:219–227, Jan 2011.
15. M. Wooldridge. *MultiAgent Systems.* John Wiley and Sons, 2002.