Towards Distributed Agent Environments for Pervasive Healthcare

Stefano Bromuri[1], Michael Ignaz Schumacher[1], and Kostas Stathis[2]

[1] Institute of Business Information Systems
University of Applied Sciences Western Switzerland (HES-SO)
TechnoArk 3, CH-3960 Sierre, Switzerland
{stefano.bromuri,michael.schumacher}@hevs.ch
[2] Department of Computer Science
Royal Holloway University of London (RHUL)
Egham, UK
kostas.stathis@rhul.ac.uk

Abstract. In this paper we present a prototypical pervasive health care infrastructure, whose purpose is the continuous monitoring of pregnant women with gestational diabetes mellitus. In this infrastructure, patients are equipped with a body-area network made of sensors to control blood pressure and glucose levels, where the sensors are connected to a smart phone working as a hub to collect the data. These data is then fed to a pervasive GRID where abductive agents provide a diagnosis for the actual reading of the sensors and contacting health care professionals if necessary. We also show how, by applying the concept of agent environment, we are facilitated in defining a pervasive GRID for roaming agents that monitor continuously the health status of the patients.

1 Introduction

If current trends in mobile phone technologies, personal digital assistants, and wireless networking are indicative of the way people will interact between them in the future, then our everyday activities is likely to be based upon an abundance of devices and applications providing the computational resources of a complex ubiquitous computing environment. Although the potential of combining these numerous applications and devices is very promising, many different current applications leave the environment's functionalities unexplored and only a small fraction of the environments potential is utilised.

Of particular relevance to the intelligent environment area is the problem of healthcare monitoring using ICT. As cures for life threatening conditions are being discovered, life expectancy increases significantly. As a consequence, the cost for healthcare will grow significantly due to the rise of the number of people that have permanent or chronic health conditions.

Diabetes is a very common chronic illness that is the fourth leading cause of death in most developed countries [1]. Amongst all the conditions related to diabetes, the case of gestational diabetes mellitus (GDM) is of particular interest

J. Dix and C. Witteveen (Eds.): MATES 2010, LNAI 6251, pp. 125–137, 2010.

as it occurs during pregnancy due to increased resistance to insulin but the precise mechanisms underlying it remain unknown. About 4% of pregnant women incur in this sort of complication. The current approach includes a planned diet, exercise and self-blood glucose monitoring tests that can be administered at home. In several cases the doctor requires that the patient visits the dietitian twice per week. However, often two checks every week are not enough: as if the hyperglycemia last for more than one day, this may cause macrosomia (excessive growth of the foetus). Thus, in these cases it is important to act as fast as possible to prevent any serious complication to the mother and the baby, by normalising the blood pressure and glucose levels with appropriate and quick treatments.

To achieve continuous monitoring, we propose a prototypical pervasive healthcare infrastructure, to collect data, monitor and alert GDM patients and inform their caretakers with historical values. Our primary goal is to break the boundaries of the hospital care, allowing patients to be monitored while living their day-to-day life and to keep in touch with healthcare professionals. The importance and significance of the proposed study is to show how by using the mechanisms proposed by distributed agent environments, we can model a distributed infrastructure to provide continuous monitoring to GDM affected women, by means of situated cognitive agents programmed using abductive logic programming. It is important to say that the focus of this study is not on creating a new cognitive model for agents, rather than showing how to model pervasive healthcare applications by means of distributed agent environments. In particular, this paper proposes an early stage prototype of the infrastructure. A first in-lab prototype has been developed, defining the agents functionalities, running simulators of sensors, using Android [9] and testing on single computer. The evaluation of the prototype in real settings and its extensions are subject of future work.

The reminder of the paper is structured as follows: Section 2 presents a background on the problem of GDM; Section 3 is a description of the system we developed; Section 4 presents the relevant related work; finally Section 5 concludes this paper and draws the lines for future work.

2 Motivating Scenario: Gestational Diabetes Mellitus

During pregnancy, some women have such high levels of glucose in their blood that their body cannot produce enough insulin to absorb it all [15]. As specified in [18], GDM affects approximately 4% of pregnant women. GDM is frequently associated with age, pregnancy weight, family history and ethnicity. GDM can increase the risk of health problems developing in an unborn baby, so it is important that the glucose levels in the pregnant woman blood are under control. If untreated or poorly controlled, GDM can cause the baby to: have macrosomia (excessive weight at birth); develop hypoglycemia at birth; develop jaundice (yellow skin); develop respiratory distress syndrome; die after week 28 of pregnancy; die in infancy. As Van Wootten and Turner specify in [18], it is estimated that

in normal pregnancies the rate of macrosomia occurrence is about 10%, while in pregnancies where GDM is involved, this rate is around 44% [17], but when a woman with GDM has some sort of basic nutrition counseling with glucose monitoring, this rates drops to 14%-18%.

In GDM, if the blood pressure keeps high for a long time and the patient experiences stomachache, headache or oedema, there is a high risk of preeclampsia which is a pregnancy induced hypertensive state. Preeclampsia may develop from the 20th week of gestation and it is characterised by high blood pressure and about 300 mg of proteins in the urine in a 24h sample, a condition called protenuria. Preeclampsia is different from a condition called Pregnancy Induced Hypertension (PIH), which involves developing high blood pressure without protenuria. Preeclampsia is generally asymptomatic but it may evolve to eclampsia, a life-threatening complication characterised by seizures and eventually coma or death. Both preeclampsia and PIH are considered very serious conditions to keep under control [12].

It is clear that in the scenario of GDM, having a system that allows for continuous monitoring would be of great benefit to reduce the rates of macrosomia in women affected by GDM and to reduce the risks of preeclampsia, but there are some assumptions that is necessary to make and some requirements that it is necessary to consider: (a) pregnant women are usually young and they are used to technologies such as smart phones; (b) pregnant women want to maintain their lifestyle and carry on their day-to-day activities; (c) GDM is characterised by blood pressure, glycemia, body weight and a set of symptoms and complications that are inter-related between each other.

3 The Pervasive Healthcare Infrastructure

As the basis for the definition of our prototype, we utilised a pre-existing distributed agent platform called GOLEM [3,4,6,5] which is based on the concept of agent environment. First of all it is better to specify what we mean with *agent environment* and *environment*. In general with the term *environment* we mean the world that is external to the agents and that the agents can inspect by using the *agent environment*. On one hand we define the *agent environment* as an entity that mediates the interaction between the agents and resources deployed in the system, working as medium of interaction. On the other hand the *agent environment* hides to the agents the complexity of dealing with the state of the environment, by providing standard interfaces and standard descriptions to the resources in the external environment. In the scope of this paper we use *environment* in terms of a place or a set of places delimited by borders defined in terms of longitude and latitude in the real environment, and that are mapped to a *distributed agent environment* for monitoring purposes, where every node of the distributed agent environment has an assigned area of the real environment.

As specified in [4], the GOLEM platform models a distributed environment in terms of the patterns proposed by distributed event based systems (DEBS) [2]. Thus, the advantage of GOLEM is that we can utilise a platform that handle the

dispatching of events in a distributed environment where the entities deployed on top of the platform are publishers and subscribers of events. GOLEM models four main entities which are objects, agents, avatars and containers.

Objects are passive reactive entities that encapsulate a service. Such objects can be used by *agents*, which represent the cognitive part of the agent environment. *Avatars* are particular kind of agents that represent users in the agent environment. Agents, objects and avatars are deployed within *containers*. Containers represent a portion of the distributed agent environment and they work as mediators for the interaction happening between agents and objects in the distributed settings. The container behaviour in GOLEM can be defined declaratively by means of the Ambient Event Calculus (AEC). The AEC is a particular dialect of Event Calculus [11] that allows to handle the concurrent modification of objects states and agent states in distributed settings.

The AEC also allows to define topologies of GOLEM containers in terms of neighbours containers and super and sub containers. The purpose of the pervasive healthcare environment we deployed is to deliver continuous monitoring to GDM patients outside the boundaries of hospital care, allowing healthcare professionals to keep in touch with the patients and allowing the patients to keep their life style. The pervasive healthcare infrastructure is associated to a real environment where the patients can move using a mobile phone to connect to GOLEM containers in form of avatars. Every GOLEM container in the network represents a different area of the city where the patient resides. For example Fig. 1(a) shows a portion of a network associated to an area of the city of Lausanne, around the Centre Hospitalier Universitaire Vaudois (CHUV).

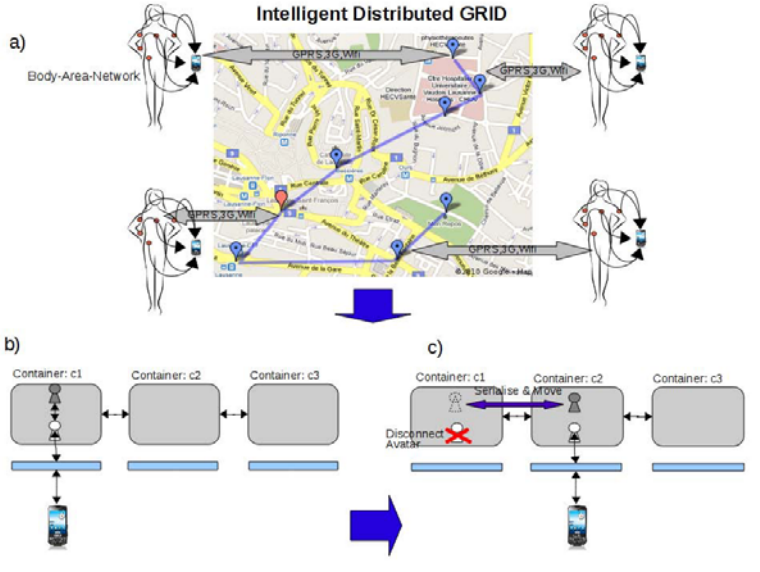

Fig. 1. a) Intelligent Environment fo Diabetes Monitoring b) and c) show how mobility of avatars and agents take place due to the mobile phone changing location in the real environment

Utilising an agent environment to communicate with a smartphone brings a set of advantages with respect to smartphone-only based solutions. First of all despite the fact that modern smart phones have a lot of computational power, computation and communication intensive applications tend to consume their battery very quickly, while in our case we only have to deal with communication with the network. Secondly, having an agent environment at support of the application allows us to introduce new and personalised services at runtime, decoupling the analysis of the data from its production. Thirdly, we can decouple the reasoning, embedded in the agents, from the actual services, embedded in the objects available to the agents in a particular location. Finally, modelling the pervasive healthcare environment as a distributed agent environment, allows us to reuse the mediation capabilities of the agent environment to define coordination and communication patterns between the agents when needed, while with a hub only based solution this kind of interaction would be technically difficult to support.

To represent the state of a GOLEM container, such as the ones shown in Fig. 1(b) and Fig. 1(c) we use the object-based notation of C-logic, a formalism that describes objects as complex terms that have a straightforward translation to first-order logic [7] and can be queried using the AEC. For example the following C-logic term specifies that

```
pervasive_golem_node:c1[
 uri ⇒ "container://one@134.219.7.1:13000", type ⇒ open,
 latitude ⇒ 46.5253, longitude ⇒ 6.6438,
 location_name ⇒ 'Centre Hopitalier Universitaire Vaudois Lausanne',
 neighbours ⇒ { pervasive_golem_node:c2, pervasive_golem_node:c3},
 entities ⇒ { agent:a1, agent:a2}]
```

a GOLEM container c1 has been deployed, it is identified by the URI container://one@134.219.7.1:13000, is an open container, it is associated with a certain latitude and longitude in the real environment, it represents the location named 'Centre Hopitalier Universitaire Vaudois Lausanne' and it has a set of neighbours in the distributed agent environment. To deal with the distributed topology presented in Fig. 1(a) we use the predicates of the AEC. For example, the following two AEC rules (see [4] for a more detailed description):

```
happens(Event,T)← attempt(Event,T), possible(Event,T).
happens(Event, T)← attempt(Event, T), necessary(Event, T).
```

specify that an action in the GOLEM agent environment happens only if it has been attempted and it is possible or necessary, where possible/2 and necessary/2 rules are application dependent rules. In other words, possible/2 rules specify what are the actions that is possible to perform in the environment given its current (possibly distributed state), while the necessary/2 rules specify what are the actions that happens as a consequence to previous events.

Moreover, to provide the mediation necessary to handle events in the distributed setting in [4] we presented the locally_at/8, neighbouring_at/9 and regionally_at/9 primitive predicates to link the state of distributed containers, following a logic programming approach. Briefly, the definition of locally_at is as follows:

```
locally_at(CId, Path, Path*, Id, Cls, Att, V, T)←
  holds_at(CId, container, entity_of, Id, T),
  holds_at(Id, Cls, Att, V, T),
  append(Path, [CId], Path*).

locally_at(CId, Path, Path*, Id, Cls, Att, V, T)←
    instance_of(SCId, container, T),
    holds_at(SCId, container, super, CId, T),
    append(Path, [CId], NewPath),
    locally_at(SCId, NewPath, Path*, Id, Cls, Att,V,T).
```

The definition of locally_at/8 states that the state of an entity can be inferred either from the top-level container or from a sub-container. If the states is inferred in the top-level container, then the predicate holds_at/5 is applied to infer the attribute Att of value V of an entity of class Cls and identifier Id. If the first predicate fails, then the second predicate moves the computation in a sub-container. In this way containers can be recursively embedded inside other containers as objects, according to the topology needed, and deployed on different hosts. The neighbouring_at/9 and the regionally_at/9 predicates have a similar behaviour but allow to query adjacent and super-containers respectively. Finally, to specify how the state of an entity modifies over time, we utilise initiates/5 and terminates/5 rules. For example, the following initiates/5 rule specifies when the position of an agent changes to the one the agent moves to:

```
initiates(E, avatar, A, position, Pos) ← do:E [actor ⇒ A, act ⇒ move:M [destination⇒ Pos]].
```

The complete description of the event's effects also requires to terminate the attribute holding the old position of the agent by means of a terminate/5 rule. In the current prototype the agent environment takes care of pairing agents and avatars as well as defining the mobility rules (i.e. what are the conditions that move an agent from one container to another) that implement the behaviour shown in Fig. 1(b) and Fig. 1(c). We define the following rules for mobility purposes:

```
possible(E,T)←
   move:E[actor⇒avatar:A, move ⇒ Pos],
   instance_of(Id,topology,T),
   holds_at(Id,topology,borders,Bdr,T),
   inside_borders(Bdr, Pos).

possible(E,T)←
    instance_of(Id,topology,T),
    holds_at(Id,topology,borders,Borders,T),
    outside_borders(Bdr, Pos),
    neighbouring_at(this, [], [C], 1, Id, topology, borders, Bdr, T),
    inside_borders(Bdr,Pos).
```

The first one states that it is possible to move in the space represented by a container only if this space is within the borders controlled by the container. Otherwise, the second rule specifies that it is possible to move outside the borders only if there is another container that is responsible for a certain area where the patient is currently moving. The following AEC rules:

```
necessary(E, T)←
   happens(E*, T),
   deploy:E*[deploy⇒avatar:Av],
   not neighbouring_at(this, [], [C], 1, Av, caretaker, _, T),
   deploy:E[agent⇒caretaker:A].

necessary(E, T)←
     happens(E*, T),
     disconnect:E*[actor⇒A, new_container ⇒ C],
     holds_at(A,avatar,caretaker,Id,T),
     physical_act:E[move_to⇒ C agent⇒ Id].
```

State respectively that whenever an avatar is deployed in the agent environment (event E*), also its caretaker agent is deployed (event E), and that whenever an avatar disconnects from the agent environment to connect to a new container, the agent associated to the avatar is also serialised and moved to the new container. Finally, a further necessary/2 rule defines that an avatar is moved to a different container when outside the boundaries of the current container, but we omit the details as it is simpler than the ones presented above.

3.1 The Body-Area Network

In addition to objects and agents, GOLEM allows the embodiment of users by means of avatars in the distributed agent environment. In this prototype every patient is equipped with a smart phone loaded with a software capable to read the data produced by the sensors worn by the patient. The smart phone then allows the patient to interact with the GOLEM agent environment by means of their avatar as shown in Fig. 2.

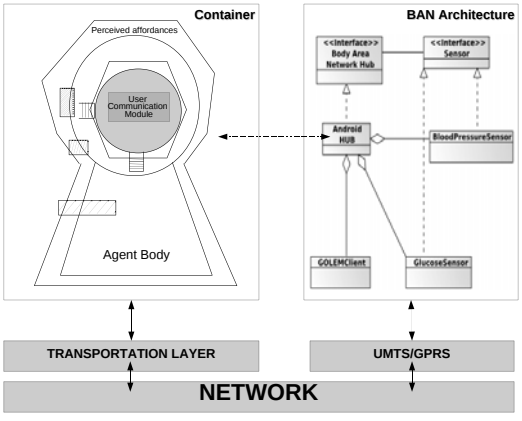

Fig. 2. The BAN Architecture

Users of the network have a wearable body-area network (BAN) that monitors periodically the blood pressure and the glucose levels of the patient. In the current prototype, the sensors are simulated and the values are entered directly by the patients, but in the future BAN will be built in term of Bluetooth sensors that monitor the physiological signs of the patient. In more details, the BAN monitors the variation in time of three values which are systolic blood pressure, diastolic blood pressure and glucose levels.

Moreover the terminal allows the patient to specify the symptoms that are being experienced during the day, through the interface shown in Fig. 3.

Thanks to the fact that the users are embodied in the agent environment as avatars, they can produce events in the agent environment as the following one:

```
pressure_reading:e1[ avatar ⇒ avid1, caretaker_agent ⇒ ag1,systolic_pressure ⇒ 120, diastolic_pressure ⇒ 80].
location:e2[avatar ⇒ avid1,latitude ⇒ Lat, longitude ⇒ Lon].
```

The event specified above is pressure_reading event with identifier e1, produced by the avatar avid1 for the caretaker agent ag1, and it contains a systolic pressure value of 120 and a diastolic pressure value of 80, while the event e2 is used by the system to keep track of the patient in the real environment and to move her from one container to another when the necessary/2 rules previously explained are triggered.

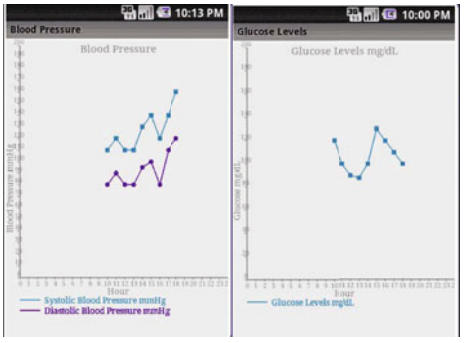

Fig. 3. The Smart Phone UI

3.2 The Caretaker Abductive Agents

In this paper we focus on agents that can diagnose the current condition of a GDM affected patient by means of abuctive logic programmed agents, differently from [6], where we focused on the navigation in the environment. In particular in this section we will describe the current prototypical cognitive model, exemplifying the behavior of the agent given a particular situation or event by showing extracts of the agent mind code.

Abductive logic programming (ALP) is a high level knowledge-representation framework that can be used to solve problems declaratively based on the idea that a set of seemingly unrelated observed facts (results), are somehow connected according to well known laws, thus offering an explanation of what might be true. As defined in [10], given a background theory T, and an observation G, the task of ALP is to compute a set of ground atoms Δ called explanation, and a ground substitution θ such that $\Delta \cup T \models G\theta$. Moreover, the set of atoms contained in Δ belongs to a set of predicates A, also called abducibles that are predicates for which there is not complete information. More formally we can say that an abductive framework is expressed in terms of a tuple $< T, \Delta, IC >$ where T is a knowledge base, Δ a set of abducibles and IC a set of integrity constraints on the abducibles. We utilise the abductive locic agent mind architecture depicted in Fig. 4.

Such an agent mind is based on the following cycle, which is an extension of the model presented in [5]:

```
cycle(T)← see(Percept, T), revise(Percept, T), choose(Action, T), execute(Action, T), now(Tn), cycle(Tn).
```

Briefly, the see/2 stage takes a percept out of the queue of percepts at a certain time T and it passes it to a revise/2 stage which in turns updates an event calculus database keeping the state of the world that is of interest for the agent (in this case the patient status). The most important stage is the choose/2 stage, of which we show the specification below:

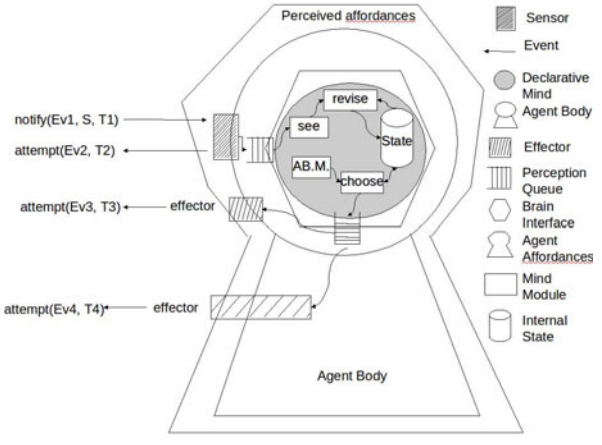

Fig. 4. Abductive Agent Mind Cycle

```
choose(Action, T)←
    instance_of(AvatarID, avatar, T),
    findall(S, holds_at(AvatarID,symptom,S,T), Symptoms),
    findall(A, select(Symptoms,A,T), Acts),
    higher_priority(Acts, Action, T).
```

```
higher_priority(ActList, Act, T)←
    member(Act, ActList), priority(Act, P, T),
    not (member(ActX, ActList), not ActX = Act,
    priority(ActX, PX,T),
    PX > P).
```

The choose stage selects a possible action by means of the select/3 and higher_priority/3 predicates. The select/3 predicate chooses then the best action to perform (such as contacting a doctor), given a diagnosis produced by the abductive module. For example a subset of the rules for GDM within the abductive module of the agent mind, can formalised as follows:

$DomainKnowledge$:

```
oedema←preeclampsia(yes), protenuria(yes).

blood_pressure(S,D)← preeclampsia(yes), protenuria(yes), pih(no), sys(160, S, 240), dias(100, D,150).
blood_pressure(S,D)← preeclampsia(no), protenuria(no), pih(yes), sys(160, S, 240), dias(100, D,150).

glucose(G)← macrosomia(yes), G > 150.
glucose(G)← hypoglicemia(yes), G < 80.
bmi(BMI)← macrosomia(yes), BMI >30.
```

IC :

```
← preeclampsia(yes), protenuria(no).
← preeclampsia(yes), pih(yes).
← pih(yes), protenuria(yes).
```

where the head of the rules in the domain knowledge represents the symptoms observed, while the body represents the abducible predicates that are part of the explanation associated to the symptoms observed.

The select/3 rules define the best action to take given a certain diagnosis produced by the demo/2 predicate, that queries the abductive module. The implementation of demo/2 predicate is based on Prologica [13] and it takes the

symptoms, which correspond to the observations set G in ALP, to find an explanation, that corresponds to Δ in ALP. The knowledge base T of ALP is implicitly represented by the domain knowledge and the integrity constraints queried by the demo/2 predicate. For instance, we can define the following select/3 rule for the case when pre-eclampsia is diagnosed with very high blood pressure (which means that there is a high risk of eclampsia):

```
select(Symptoms, A, T)←
        demo(Symptoms, Explanation),
        M = m[diagnosis ⇒ eclampsia, diastolic ⇒ D, systolic ⇒ S, patient ⇒ ID, location ⇒ Loc],
        A = email:act1[actor ⇒ AID,doctor_email ⇒ DE, message ⇒ M, priority ⇒ 10],
        subset_of([preeclampsia(yes),sys(160, S, 240), dias(100, D,150)], Explanation),
        instance_of(ID,patient, T), holds_at(ID, doctor_email, DE, T),
        holds_at(ID, current_location, Loc, T),
        myID(AID).
```

The rule above specifies that, given a diagnosis of pre-eclampsia with high blood pressure (D stands for diastolic, S stands for systolic), the action to perform is to send an email to the current doctor of the patient for which the agent is in charge, specifying the current diagnosis.

4 Related Work

There have been several attempts to deal with the issues here presented. For example, Schaeffer-Filho et al. [14] define the concept of *Self Managed Cell* (SMC). Schaeffer-Filho defined the concept of SMC as a recursive structure that goes from the body-area network for health monitoring of the patient to the SMC to handle the household of the patient to the SMC of the healthcare professionals in charge of the patient. Such SMCs are structured with an event bus designed to follow the publisher/subscriber pattern [2]. The BAN is modelled as a virtual complex node that abstracts a set of sensors and publish events in the form of health records in the upper level SMC. The doctor SMC works as a subscriber for the events produced by the BAN. Thus, it could happen that the events published by the SMC are then retrieved by the doctor SMC to have a better view on the condition of the patient. From a certain perspective we can relate the concept of SMC to the concept of agent environment. As shown in our prototype, SMCs are based on DEBS patterns for the notification and dispatching of events. The most relevant difference between the approach proposed by SMCs and ours is that we utilise cognitive agents to prefilter the data produced by the BAN and that we model the behaviour of the agent environment as a declarative structure that evolves over time.

Another attempt to model an infrastructure for pervasive healthcare is presented by Wagner and Nielsen in [16]. Wagner and Nielsen envision an architecture based on 4 logic tiers: Public Tier, Central Tier, Home Tier and Mobile Tier. The Public Tier is publicly available as a SOA-based infrastructure which comprises a set of services for professional caretakers, nurses and doctors. The central tier models the domain model, specifying how the data are exchanged/accessed

by the various actors of the system. The Home Tier is represented as a touch screen available in every house that the patient can use and a Mobile Tier take care of those situations when the patient leaves his house. With respect to OpenCare, our infrastructure embeds cognitive agents that can be programmed to monitor the patients according to the condition affecting them. Moreover, the use of the GOLEM infrastructure allows us to define a clear topology for the environment, which is missing from the OpenCare project.

In [8] Ciampolini et al. present a distributed MAS based on the ALIAS language to deal with distributed diagnosis. The agents of such a system are programmed according to the principles of abductive logic programming. Of particular relevance is the fact that Ciampolini et al. define a procedure to combine the results of different diagnosis produced by multiple expert agents in a single combined diagnosis. Moreover, in Ciampolini's approach the diagnosis is provided in term of probabilities. With respect to the work proposed by Ciampolini et al. we have a simpler reasoning procedure as we do not combine the diagnosis proposed by multiple agents. The contribution of our prototype with respect to the work proposed by Ciampolini is the introduction of the agent environment to foster continuous monitoring of the patients and to coordinate the interaction between the user and the agent environment to allow the agent to produce complex actions, such as sending an alert to the patient and to healthcare professionals when needed.

5 Conclusion

In this paper we presented a novel prototypical pervasive healthcare infrastructure to monitor patients affected by GDM in their day-to-day activities. The protype is defined in terms of a Body-Area Network based on a smart phone and on a set of sensors to check the physiological signs of a patients. Such physiological signs are then sent as events to the GOLEM agent infrastructure where roaming mobile agents, capable to perform abductive reasoning, check the events produced in the agent environment. Thus, if a critical situation occurs, the agents notify the healthcare professionals in charge of the patient.

One of the advantages of our simulated infrastructure using the GOLEM and Android mobile phone platforms is that it can be deployed in a real setting by further extending the knowledge of the agents, the topology of the containers, and the body sensor network functionalities. In such a setting, an important issue is how to store and retrieve the information produced in the pervasive healthcare agent environment. Such information is of medical importance as the data retrieved could help to uncover unknown patterns of certain illnesses, thus storing it in an appropriate manner is an important direction for future work. The next steps of the project include: (a) substitute the simulated sensors with real ones; (b) deploy the infrastructure in real settings; (c) evaluate the platform with a pilot study at CHUV.

Acknowledgement

This work was partially supported by the COST Action on Agreement Technologies. The authors would like to thank Dr Ruiz and the Department of Endocrinology and Diabetes at CHUV for the support during the definition of the prototype.

References

1. The effect of intensive treatment of diabetes on the development and progression of long-term complications in insulin-dependent diabetes mellitus. The Diabetes Control and Complications Trial Research Group. N. Engl. J. Med. 329, 977–986 (September 1993)
2. Blanco, R., Wang, J., Alencar, P.: A Metamodel for Distributed Event-based Systems. In: DEBS '08: Proceedings of the Second International Conference on Distributed Event-Based Systems, pp. 221–232. ACM, New York (2008)
3. Bromuri, S., Stathis, K.: Situating Cognitive Agents in GOLEM. In: Weyns, D., Brueckner, S.A., Demazeau, Y. (eds.) EEMMAS 2007. LNCS (LNAI), vol. 5049, pp. 115–134. Springer, Heidelberg (2008)
4. Bromuri, S., Stathis, K.: Distributed Agent Environments in the Ambient Event Calculus. In: DEBS '09: Proceedings of the Third International Conference on Distributed Event-Based Systems. ACM, New York (2009)
5. Bromuri, S., Urovi, V., Stathis, K.: Game-based E-retailing in Golem Agent Environments. Journal of Pervasive and Mobile Computing 5(4) (2009) (in Press)
6. Bromuri, S., Urovi, V., Stathis, K.: iCampus: A Connected Campus in the Ambient Event Calculus. International Journal of Ambient Computing and Intelligence 2(1), 59–65 (2010)
7. Chen, W., Warren, D.S.: C-logic of Complex Objects. In: PODS '89: Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 369–378. ACM Press, New York (1989)
8. Ciampolini, A., Mello, P., Storari, S.: Distributed medical diagnosis with abductive logic agents. In: ECAI 2002 Workshop on Agents in Healthcare, Lione (2002)
9. Google Inc. What is Android? (2008), Home Page, `http://code.google.com/android/what-is-android.html`
10. Kakas, A.C., Kowalski, R.A., Toni, F.: Abductive logic programming. J. Log. Comput. 2(6), 719–770 (1992)
11. Kowalski, R., Sergot, M.: A logic-based calculus of events. New Gen. Comput. 4(1), 67–95 (1986)
12. Petit, P., Top, M., Chantraine, F., Brichant, J.F., Dewandre, P.Y., Foidart, J.M.: Treatment of severe preeclampsia: until when and for what risks/benefits? Rev. Med. Liege. 64, 620–625 (2009)
13. Ray, O., Kakas, A.: Prologica: a practical system for abductive logic programming. In: Dix, J., Hunter, A. (eds.) 11th International Workshop on Non-Monotonic Reasoning, pp. 304–312 (May 2006)
14. Schaeffer-Filho, A., Lupu, E., Sloman, M.: Realising management and composition of self-managed cells in pervasive healthcare, pp. 1–8 (April 2009)

15. Serlin, D.C., Lash, R.W.: Diagnosis and management of gestational diabetes mellitus. Am. Fam. Physician 80, 57–62 (2009)
16. Wagner, S., Nielsen, C.: OpenCare project: An open, flexible and easily extendible infrastructure for pervasive healthcare assisted living solutions, pp. 1–10 (April 2009)
17. Warren, J.M.: Pregnancy outcomes in women with gestational diabetes compared with the general obstetric population. Obstet. Gynecol. 91, 638–639 (1998)
18. Van Wootten, W., Elaine Turner, R.: Macrosomia in neonates of mothers with gestational diabetes is associated with body mass index and previous gestational diabetes. Journal of the American Dietetic Association 102(2), 241–243 (2002)