Noname manuscript No.
(will be inserted by the editor)

An Agent-based approach towards Automatic Service Composition in Ambient Intelligence

First Author · Second Author

Received: date / Accepted: date

Abstract Systems for Ambient Intelligence environments involve at some stage a service composition task, as a mean of adaptability to the context changes. However, users generally find themselves involved in the composition task, by selecting or deciding what to compose and how. This paper proposes the use of Artificial Intelligent Agents for the automation of the composition task, providing transparency from the user point of view.

Keywords First keyword · Second keyword · More

1 Introduction

The Ubiquitous Computing concept was first defined by Mark Weiser in [20], referring to a new computing era where electronic devices merge with the background. People make use of these electronic devices unconsciously, focusing just on their needs and not in how to accomplish them.
The concept of Ambient Intelligence [7], lying on the ubiquitous computing paradigm, refers to those environments where people are surrounded by all kind of intelligent intuitive devices capable of recognising and responding to their changing needs. People perceive the surrounding as a service provider that satisfies their needs or inquiries in a seamless, unobtrusive and invisible way.
Traditional paradigms make some simple assumptions, such as high bandwitdh, reliable connectivity, fixed network topology or hardware capabilities. These assumptions do not correspond themself to the reality of an Ambient Intelligent environment, mainly characterized by its dynamism and the existence heterogeneous devices. Therefore, a wide variety of new paradigms come out with the intention of providing solutions to these open matters, based on configuration files [13], reflection [18][5], event-based [14][6], tupla space based [16], web services [12], or policy based [4], just to name a few. Among these solutions, one of the most promising approaches that better fulfils the Ambient Intelligence requirements seems to be the Service-Oriented paradigm[].
The service concept is at the core of the Service-Oriented paradigm, which basically considers resources (e.g. distributed objects, agents, or basic web services) as services, in such a way that one service is created for each available resource.
This paradigm identifies two types of services, namely: basic and composite services. The former are the services directly offered by resources, while the later are composed of basic services, which result in more complex services. User needs are generally better fulfilled by composite services than by basic ones. Nevertheless, service composition is not a trivial matter and it considerably increases the complexity of the system.
Depending on the level of autonomy, service composition can be carried out in three different ways [19]: manual composition, semi-automated composition and automatic composition.
Manual and semi-automated composition expects the user to interact in the decision making task involved in composition, that is, the user knows the available services and how those can be composed to synthesize new services. Even in semi-automated composition, the

F. Author
first address
Tel.: +123-45-678910
Fax: +123-45-678910
E-mail: fauthor@example.com

S. Author
second address

user is involved in the composition task, by selecting the services to be composed from a shortlist.
At a differente level, in Ambient Intelligent environments, the role played by the middleware architecture is essential in simplifying and abstracting the complexity and heterogeneity of both, network and device technologies. However, middlewares for traditional environments made some simple assumptions, such as high bandwidth, reliable connectivity, fixed network topology or hardware capabilities, that do not correspond themself to the reality of an Ambient Intelligence environment.
The proposal presented in this paper, suggests the use of intelligent agents, as a constituent part of the middleware architecture, for automation of the service composition task. The composer agent relieves the user from having any expertise regarding the details involved in the composition. In this way a more intelligent ambient is achieved, since the agent is in charge of knowing what, how, and when to compose the basic resource-oriented services.
The remainder of this paper is structured as follows. First the background section provides the basis of the contributions presented in this paper. The next section aims at drawing the outline of a composer agent, from the process of service discovery to the automatic service composition. The section following is devoted to provide implementation details of how to implement the ideas here outlined. Finally, the last section presents the conclusions and provides suggestions for future works.

2 Background

Automatic service composition in a dynamic context entails a high degree of adaptability and reasoning in order to react to changes, new scenarios and events. Uncertainity arises due to unforeseen situations or people demands. Therefore, Ambient Intelligence systems have to decide the response to such situations, in the same way a person would do, by gaining knowledge of the context, stablishing goals and employing reasoning in order to accomplish them.
It has to be remarked that the ability to replicate human reasoning remains one of the main objectives in the Artificial Intelligence field. In this regards, Intelligent agents have come to be a powerful solution to this issue.
Different approaches for agents bring about different agent implementations, as referred in [21]: Logic based architectures (deductive agents), reactive architecture (reactive agents) , layered architectures (hybrid agents), and practical reasoning architectures (Belief-Desire-Intention agents). Among these alternatives, the Belief-Desire-Intention model (BDI) has proved to be a powerful framework for building rational agents [21].
The BDI model of decision making is intended to reproduce the process carried out when people take decisions to achieve a certain goal.
The main characteristic of the BDI model lies in the significance conceded to beliefs, desires, and intentions involved in rational actions. Therefore, those systems that grant importance to these attitudes over any other, are often referred to as BDI-architectures.
Beliefs are the information agents hold about the world, which is not necessarily accurate. This information might change as a result of new perceptions or the execution of intentions. Desires or goals refer to those tasks that, in an ideal world, the agent would like to accomplish. Desires are to be consistent among themselves. Intentions are those desires that agents are committed to accomplish. Despite of desires consistency, the agent might not be able to accomplish all of them.
The reasoning capability is highly dependent on the semantic model used to describe the agent context, and so, providing a good semantic model has long been an issue of concern. In [8] the lack of agreement on the knowledge properties is identified as one of the main shortages that need to be overcome. J. Hintikka in [10] provided a good approach to this matter with the *possible-worlds semantics*. This semantic describes the agent's possible worlds or states accessible from its current state of the world. Rao and Georgeff proposed in [2] an approach to model possible-worlds, by means of a temporal structure with a branching time future and a single past, called time tree. CTL, CTL*, and LORA, just to name a few, are some of the formalisms used for this purpose.
Jadex [17] is an agent-oriented reasoning engine supporting the development of rational agents. In spite of using formal logic descriptions, Jadex proposes the use of two commonly known languages, such as Java and XML. The BDI agent is modelled by mapping the concepts of beliefs into Java objects, while desires and intentions are mapped into procedural recipes coded in Java that the agent carries out in order to achieve a goal.

2.1 Service Composition Requirements

The development and deployment of applications for Ambient Intelligence strongly depend on middleware architectures. This simplifies and hides the complexity of handling different network protocols, or heterogenous

devices. Therefore, it is necessary to find a way to evaluate the suitability of the different middleware architecture approaches. Obviously, a middleware architecture aimed at supporting static environments cannot cope with the dynamism or heterogeneity of an Ambient Intelligence environment. It is listed underneath the set of expected requirements for a middleware architecture in the context of Ambient Intelligence, from the point of view of service composition support:

- **Basic services:** The composition process uses basic services as its raw elements. Therefore, a middleware architecture should provide a set of basic services from which more complex services could be composed.
- **Service Discovery Protocol:** This protocol is in charge of discovering the available services along the time. The discovery task is essential for composition, since available services at a certain moment might change their state, or new ones may arise.
- **Automatic binding:** Composition involves several tasks, such as service discovery, integration of different technologies, or satisfaction of dynamic service dependencies. All these tasks have to be automatically accomplished.
- **Uniform treatment of services:** Despite the fact that services might be offered by different sources, all of them have to present the same structure, as if they come from an unique source.
- **External automatic service adaptation:** At some point it can be necessary to carry out an adaptation process during the composition, since external source services may differ in regards of the interaction models and protocols. It is necessary to mask out these differences by means of translation and interface masks.
- **Dynamic service ranking:** More than one service can be suitable for a request, although just one is selected. It is necessary to effectively evaluate the suitability of one over the others. This evaluation is dynamic, which means that available services are in continuous evaluation.
- **Service continuity:** Services that take part in a composition are not fixed. A service selected for a composition might be discarded in favour of a new service, which has achieved a higher mark in the ranking.

3 The Composer Agent: Implementation details

Composability has become an appealing field, mainly because of its support to Ambient Intelligence environments that dynamically react to new situations, events, or user requirements. Most of these new emerging scenarios were not taken into account at implementation time, due to the unmanageable number of combinations. This fact, along with the hetereogenity of device and network technology, makes dynamic and automatic service composition a key requirement for any middleware aimed at supporting Ambient Intelligence environments.

In spite of the attempts made to inyect some sort of dynamism into the middleware [13][18][5][14][6][16][12][4], none of them has suceed in the task of responding to new or changing context requirements, mainly due to their lack of reasoning capabilities.

This paper proposes the use of intelligent agents as an innovative idea to provide reasoning capabilities to the middleware. Nevertheless, the intention of this work consists on studying the suitability of an agent-based solution, so it is out of the scope of this paper a deep discussion about strengths and weaknesses of the different general purpose frameworks for BDI agents, as [1][3][11][15]. Among these approaches, Jadex stands out for its modularity design, use of XML and Java for generating the agents, and the ready-to-use framework that can be adapted to work on top of a tailormade middleware. These strenghts makes the use of Jadex as the most suitable choice for the objectives of this work.

Regarding the implementation details, the **composer agent**, as a BDI agent, is composed of beliefs, desires, and intentions, which in the Jadex platform are mapped into beliefs, goals, and plans. Beliefs reflect the knowledge the agent maintain about its environment. However, as stated before, this knowledge is not static but dynamic, and as so, the agent has to collect all the events raising in the environment, in order to upgrade the beliefs base when required. Beliefs determine the plans that need to be carried out in order to achieve the desired states. Plans are the recipes that determine the basic agent behaviours. The execution of basic plans leads the agent to accomplish its desired states. Nevertheless, the composer agent is not an isolated agent, but it is composed of specilized agents.

The next case of study focuses on highlighting the strenghts of the agent-based solution here proposed. Therefore, it considers an Ambient Intelligent system in charge of providing the environment with a set of different purpose services. Among these services, three of them are marked out: the *presence sensor service*, aimed at monitoring people presence in a specific room, the *camera server service*, providing control to the different videocamera devices, and finally, the *face detector service*, which extract people faces from pictures.

Avoiding the overwhelming details of such a case of study allows to focus on the important spots. Hence,

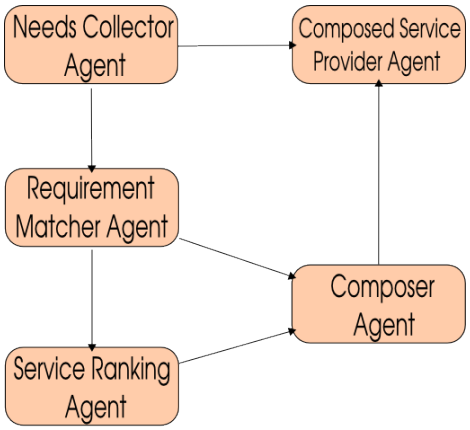

Fig. 1 The composer agent system

consider now, that due to an unknown reason, the presence sensor service becomes unavailable. However, not until a request for this service arises do the system catch an exception for a unanswered request. This exception is caught by the composer agent, which processes it, and give a response to it. This time, the response consists on snapshoting the room by means of the camera server, since the agent has identify the presence of videocamera. Aftewards, this snapshot is passed to the face detector service, intended to identify a face on it. People presence is concluded depending on the face detector results.

As stated above, although refered as composer agent, it is composed of several small entity agent. This case of study assigns a set of responsabilities to the composer agent, mainly aimed at responding to context changes by means of automatic service compositions. On the one hand, figure 2 depicts the BDI agents that composed the multi-agent system, here refered as the composer agent. These small entity agents are related among themselve, sharing not only beliefs, but plans or goals. On the other hand, figure **??** enumerate the beliefs, desires and intentions that describe the *Need Collector Agent (NCA)*. Furthermore, the next fragment of code is the *Agent Description File (ADF)* of the NCA.

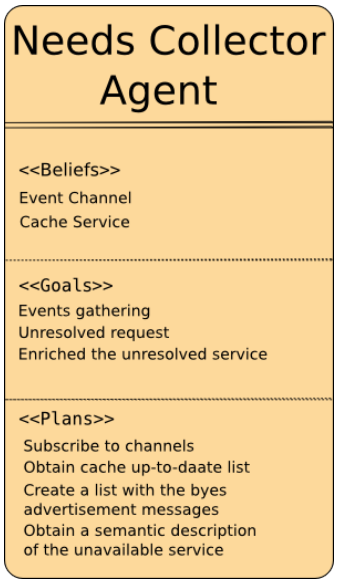

Fig. 2 NCA beliefs, goals and intentions

```
<beliefs>

  <!--- Proxy to the camera server object. At the moment is
    hard-coded but this belief is to
    be automatically retrieved using the property server--->

  <belief name="cameraServerPrx"  class="Ice.ObjectPrx">
    <fact>communicator().stringToProxy("AXIS.161.67.38.55
        -t @ AXIS.161.67.38.55") </fact>
  </belief>

  <!--- The presence sensor information --->
  <belief name="presenceSensorPrx"  class="Ice.ObjectPrx">
    <fact>communicator().stringToProxy("presenceSensor -t:tcp -h 161.67.27.229 -p 34021")</fact>
  </belief>
  <!-- Is it possible to compose services in order to determine whether the person is present or not -->
  <belief name="presence" class="Presence">
    <fact>new Presence()</fact>
  </belief>
  <!--- Is the person present? -->
  <belief name="presenceComposerEnd" class="boolean">
    <fact>$beliefbase.presence.getPresenceProccessStatus()</fact>
    <!-- getPresenceStatus returns true if the composing process has been effectively carried out.
  The person might be present or not, but the localization proccess has been achieved. -->
  </beliefs>
```

Fig. 3 Beliefs

```
<goals>
    <!-- get the value of the presence sensor -->
    <achievegoal name="getPresence">
      <parameter name="presenceValue" class="Presence">
  <!-- <targetcondition>$goal.presenceValue.getValue()</targetcondition>     -->
  <!-- <creationcondition></creationcondition> -->
    </achievegoal>
      <!--  -->

  </goals>
```

Fig. 4 Goals

4 Towards an architectural model for Automatic Service Composition

Some of the most promising approaches toward automatic service composition revolve around web services, ontologies or artificial intelligence planning techniques. Despite of the groundbreaking ideas provided by these approaches, none of them have so far succesfully addressed the dynamism implicit in an Ambient Intelligence environment.

Dynamism implies changes, that have to be captured in the knowledge the Ambient Intelligence system maintains about its environment, in other words, the system is to be context-aware. Nevertheless, capturing these changes is not enough when a self-sufficient behaviour is expected from the environment, intended to understand what is happening around, in order to wisely react. In this regard, this paper states the use of reasoning mechanisms as a way of dealing with these changes and the effects they cause on the environment.

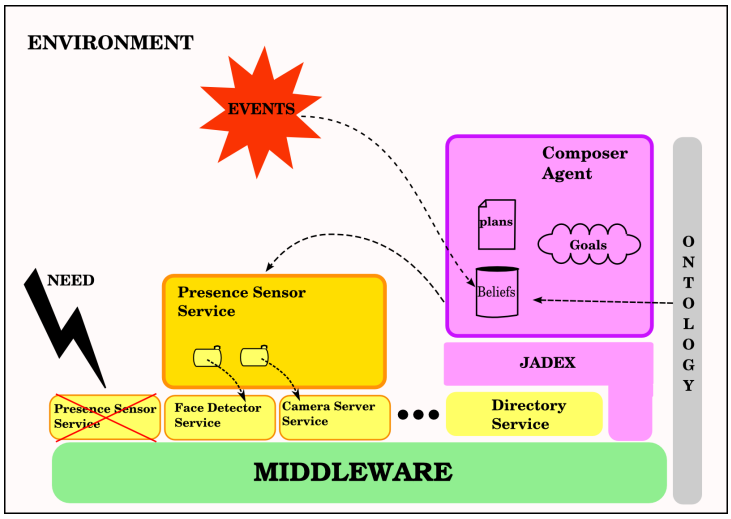

Fig. 6 Architecture

```
<plans>

  <!-- Plan for getting the camera snapshot -->
  <plan name="getSnapShot">
    <body class="getSnapShotPlan"/>
  </plan>

  <!-- Plan for getting the presence sensor status -->
  <plan name="getPresenceSensorStatus">
    <body class="getPresenceSensorStatus"/>
  </plan>

  <!--- Plan for matching the searched person -->
  <plan name="personMatching">
    <body class="personMatchinPlan"/>
  </plan>
</plans>
```

Fig. 5 Plans

The reasoning capability is at the root of the automatic service composition support. Enriching the environment with the capability to reason means that even unforeseen scenarios can be dealt by the environment, by means of the *intelligence* owned by the *ambient*.

Figure 6 depicts the architecture proposed in this paper, intended to support Ambient Intelligent environments. This proposal identifies the automatic service composition support as the main handicap to be overcome, proposing the use of intelligent agents to this mean.
The composer agent rest on top of the **middleware** layer, built using *The Internet Communications Engine* (Ice) [9]. It is an object-oriented middleware platform, similar in concept to CORBA, on top of which services are deployed. Jadex is also deployed over it, so that the middleware features can also be promoted upwards to the agent. (Despite the fact that the Ice adapter for Jadex, has not been provided yet, the well documented API and its modularity design requires little effort to develop the adapter).lo dejo?
Figure 6 captures a snapshot of the proposed architecture deployed in an Ambient Intelligence environment, where a presence sensor service was previously offered, but not available anymore. As described above, there are also some other services that can be composed into a new service behaving as a presence sensor service. The *face detection service* is capable of identifying a face from a picture, and the *camera server service* is capable of providing access and some other functionalities to any camera device, from where pictures can be taken from.
Imagine that due to an unknown error, the presence sensor server becomes unreachable. After some time, the *directory service* update its cache, getting ride In this situation, without the composer agent, the environment could not be able to face a request for a presence checking, failing in this task.
However, the composer agent here described, identifies requests for unavailable services, that answer on behalf of them, by means of the composed services. Therefore, that the combination of face detector and the camera server services can function as a presence sensor, since the camera server can be used to get a snapshot of the room, and then use the face detector to get a face from the snapshot. If the face detector fails to obtain a face, then it can be infered that the room is empty.
It can be thought that there is not a difference between what it has been exposed so far and a deep search or a planning. However, the use of an ontology is making a

difference.
So far, when a need for a service has been identified, the composer agent finds a set of services that combined together behave as the required one. If the seek for these basic services consists on finding the list of services whose outputs match the next service inputs, then it falls on the planning or deep search cathegory.
However, this proposal expects the system to behave as a person would do in the same situation, by reasoning about the context, and trying to accomplish the best action at each moment.
The gap between the composer agent behaviour and the human behaviour is mainly due to the semantic knowledge humans extract from the environment, use as the base for their reasoning. Therefore, if the composer agent manages to infer this semantic knowledge from the **ontology**, the agent will select the best plans or actions that will lead it to accomplishing its goals.
Considering figure 6, the ontology is transversally depicted in order to remark the fact that the same ontology is used by the agent platform and the middleware. Since the Jadex platform provides support for managing ontologies, the composer agent manages itself to appeal to the ontology, in order to understand what are the available services doing. Once the composer agents assign semantic meaning to these services, it just has to find out which is the intersection of services that suply the seeked functionality.

5 Conclusions and future works

Ambient Intelligence environments are characterised by high dynamism. The set of devices present in the environment is constantly in change, and consequently, the set of services offered by these devices is not fixed. Due to this dynamism, a primary identification of needs can just be used as a starting point, since new needs will arise as new devices and user requirements change along the time.
This context requires mechanisms that could effectively manage an environment constantly in change. An appropriate design of Ambient intelligence middleware, combined with service composition turn out to be the more suitable solution to this issue.
The proposal presented in this paper, exposes the use of intelligent agents for automation of the service composition task. The composer agent relieves the user from being aware of the details involved in the composition, achieving in this way a more intelligent ambient, since the agent is in charge of knowing what, how, and when to compose the basic resource-oriented services.
The intelligent agent has been modelled as a BDI Agent, using to this purpose the Jadex platform. Jadex provides a set of tools supporting the BDI Agent construction, but also it provides the reasoning engine that lead the agent into accomplishing its intentions.
The combination of the composer agent along with the middleware layer result in a middleware architecture able of tackling the dynamism inherent in Ambient Intelligence environments.
Despite the fact that this proposal suppose an important step forward real Ambient Intelligence, there is still serveral issues that have to be dealt in order to reach a full Ambient Intelligent environment. Some of these issues regard the ontology, since it is the main resource where all the semantic information is being extracted.
Furthermore, the directory service could be implemented by means of another intelligent agent, in order to provide a list of services satisfying a list of constrainst.
The main idea for this service is accomplishing a semantic search, in order to return a list of services that satisfy the restrictions stated by the composer agent.

References

1. Bordini, R.H., Hübner, J.F.: Bdi agent programming in agentspeak using *ason* (tutorial paper). In: CLIMA VI, pp. 143–164 (2005)
2. Bratman, M.E.: Intention, Plans, and Practical Reason. Harvard University Press, Cambridge, MA (1987)
3. Busetta, P., Ronnquist, R., Hodgson, A., Lucas, A.: Jack intelligent agents - components for intelligent agents in java (1999)
4. Capra, L., Emmerich, W., Mascolo, C.: Reflective middleware solutions for context-aware applications. In: REFLECTION '01: Proceedings of the Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns, pp. 126–133. Springer-Verlag, London, UK (2001)
5. Costa, P., Coulson, G., Mascolo, C., Mottola, L., Picco, G.P., Zachariadis, S.: A reconfigurable component-based middleware for networked embedded systems. Int. Journal of Wireless Information Networks **14**(2) (2007)
6. Cugola, G., Picco, G.P.: Reds: a reconfigurable dispatching system. In: SEM '06: Proceedings of the 6th international workshop on Software engineering and middleware, pp. 9–16. ACM, New York, NY, USA (2006). DOI http://doi.acm.org/10.1145/1210525.1210530
7. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.C.: Istag: Scenarios for ambient intelligence in 2010. Tech. rep., ISTAG (2001)
8. Halpern, J.Y., Moses, Y.: A guide to completeness and complexity for modal logics of knowledge and belief. Artif. Intell. **54**(3), 319–379 (1992). DOI http://dx.doi.org/10.1016/0004-3702(92)90049-4
9. Henning, M., et al.: Distributed programming with ice (2003). DOI www.zeroc.com/Ice-Manual.pdf
10. Hintikka, J.: Knowledge and Belief. Cornell University Press, Ithaca, New York (1962)
11. Huber, M.J.: Jam: a bdi-theoretic mobile agent architecture. In: AGENTS '99: Proceedings of the third

annual conference on Autonomous Agents, pp. 236–243. ACM, New York, NY, USA (1999). DOI http://doi.acm.org/10.1145/301136.301202
12. Issarny, V., Sacchetti, D., Tartanoglu, F., Sailhan, F., Chibout, R., Levy, N., Talamona, A.: Developing ambient intelligence systems: A solution based on web services. Automated Software Engg. **12**(1), 101–137 (2005). DOI http://dx.doi.org/10.1023/B:AUSE.0000049210.42738.00
13. Jo a.P.S., Garlan, D.: Aura: an architectural framework for user mobility in ubiquitous computing environments. In: WICSA 3: Proceedings of the IFIP 17th World Computer Congress - TC2 Stream / 3rd IEEE/IFIP Conference on Software Architecture, pp. 29–43. Kluwer, B.V., Deventer, The Netherlands, The Netherlands (2002)
14. Meier, R., Cahill, V.: Steam: Event-based middleware for wireless ad hoc networks. icdcsw **00**, 639 (2002). DOI http://doi.ieeecomputersociety.org/10.1109/ICDCSW.2002.1030841
15. Morley, D., Myers, K.: The spark agent framework. In: AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 714–721. IEEE Computer Society, Washington, DC, USA (2004). DOI http://dx.doi.org/10.1109/AAMAS.2004.267
16. Picco, G., et al.: Lime: A middleware for physical and logical mobility. In: ICDCS '01: Proceedings of the The 21st International Conference on Distributed Computing Systems, p. 524. IEEE Computer Society, Washington, DC, USA (2001)
17. Rao, A.S., Georgeff, M.P.: Modeling rational agents within a BDI-architecture. In: J. Allen, R. Fikes, E. Sandewall (eds.) Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91), pp. 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA (1991)
18. Veríssimo, P., Cahill, V., Casimiro, A., Cheverst, K., Friday, A., Kaiser, J.: Cortex: Towards supporting autonomous and cooperating sentient entities. In: Proceedings of European Wireless 2002, pp. 595–601. Florence, Italy (2002)
19. David W. Walker W. A. Gray, S.M.: A framework for automated service composition in service-oriented architectures. In: ESWS, pp. 269–283 (2004)
20. Weiser, M.: The computer for the 21st century pp. 933–940 (1995)
21. Wooldridge, M.J.: Reasoning about Rational Agents. The MIT Press, Cambridge, Massachusetts (2000)