CAmIE: An Agent-Based Model for the Development of Large-Scale AmI Environments

Marius-Tudor Benea[a,b], Amal El Fallah Seghrouchni[a], and Adina Magda Florea[b]

a) LIP6 - University Pierre and Marie Curie, France
b) Computer Science Department - University Politehnica of Bucharest, Romania,
{marius-tudor.benea,amal.elfallah}@lip6.fr
adina.florea@cs.pub.ro

Abstract. This work introduces CAmIE, an agent-based model for the bottom-up development of large-scale AmI environments. It offers tools to easily develop AmI applications and, while not visible to the users, it assures the interactions between these applications at runtime, enhancing them with a highly intelligent behavior. A series of distribution platforms as well as a collection of shared knowledge libraries are used, to motivate the human actors to contribute, leading, thus, to environments that integrate in the best possible way their collective intelligence.

Keywords: ambient intelligence, collective intelligence, multi-agent systems, agent-based AmI applications, large-scale AmI environments

1 Introduction

The field of ambient intelligence (AmI) is characterized mainly by middleware solutions, small-scale, scenario-based, applications and innovative user-friendly interfaces. Developing large-scale AmI environments is an ambitious goal and we lack Application layer solutions. To solve this problem, we propose the CAmIE (*Collective AmI Environments*) model. It uses an agent-based, bottom-up, approach and it aims at endowing the AmI environments with the collective intelligence (CI) that could result from enhanced interactions between their main actors. This makes the level of intelligence grow with the environment's size, facilitating the progressive development of highly intelligent AmI environments.

CI is a property of systems emerging from the interaction between their components. It is considered to be higher than the sum of intelligence of all the actors, taken separately. An important factor that influences the CI of the groups is diversity, [5]. Moreover, the most intelligent actors of an AmI environment are the human actors and they are also the best resource for evaluating it, considering the user-centered nature of ambient intelligence. Consequently, CAmIE tries to integrate, in the best possible way, the CI of the human actors (users and

several classes of developers), into the environment. A series of tools are offered to the developers with respect to this principle. A set of distribution platforms makes possible their evolution. A collection of knowledge libraries encourages the users, too, to share their knowledge. Finally, the interaction between AmI applications, made possible by the use of agent technologies, creates the premise for the environment's growth and for the emergence of CI.

The main principles behind CAmIE and a literature analysis were offered in a previous work on the CAmI language, [3], which is part of the model. The goal of this paper is, thus, to provide a brief description of its remaining components.

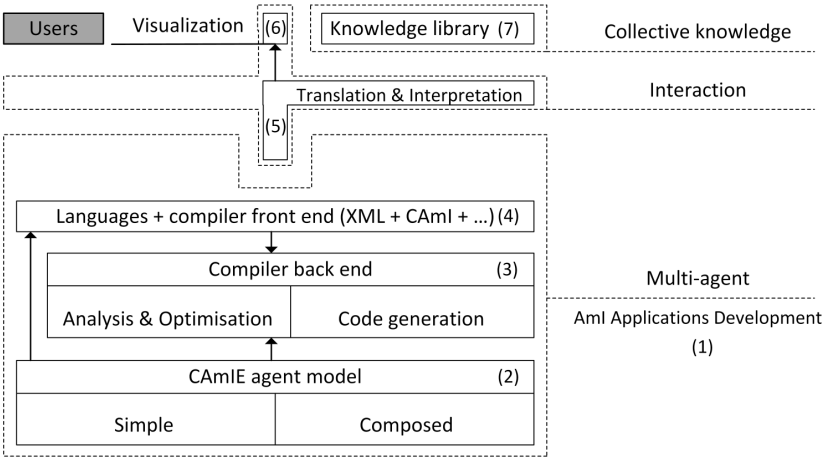

Fig. 1. Components of CAmIE associated to the development

2 The CAmIE Model

2.1 Layers

Our model regards the *Application* layer of AmI (the AmI layers are: Hardware, Interconnectivity, Interoperability, Application and Interface), responsible with the intelligent behavior of the environment. It introduces three sublayers for it (see Fig. 1 and Fig. 2): the *Multi-agent layer*, represented by the multi-agent systems executing in the environment, the *Collective knowledge layer* (CKL), [3], for sharing knowledge between the actors (see Section 2.2) of the environment and the *Interaction layer* assuring the interactions between them, between AmI systems, at the MAS level, and between the users and the CKL.

2.2 Processes and Actors

There are two processes: *development* and *execution*, which are strongly interconnected and interdependent. The interdependence is made possible by an eval-

uation and evolution guidance system that collects very simple feedback from the large community of users of such environments, in order to help improving them. The actors available in the environment are: *human actors* and *agents*. The human actors are either *developers* or *users*.

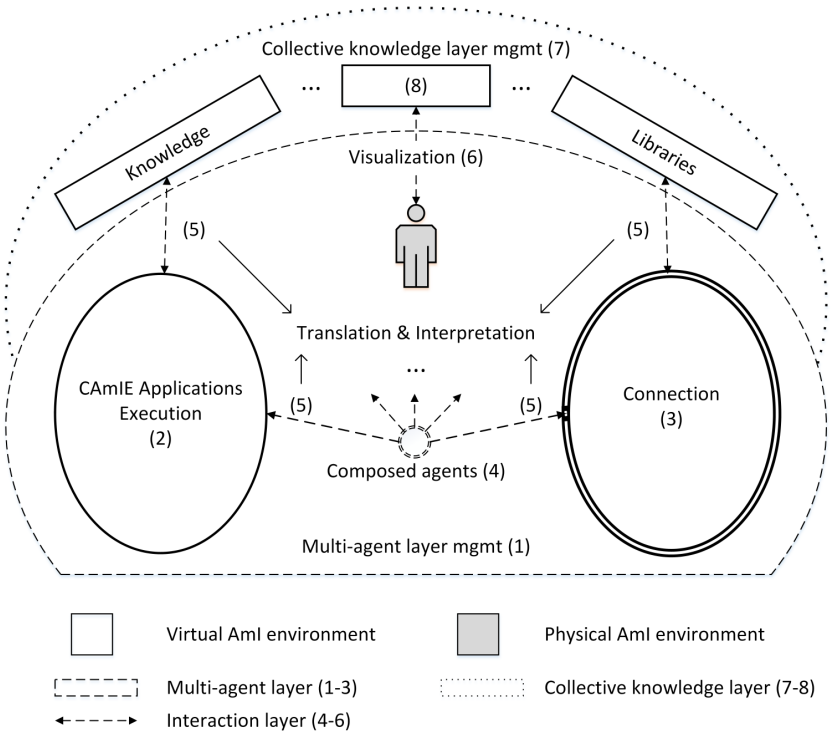

Fig. 2. Components of CAmIE associated to the execution (ellipses – AmI systems)

2.3 The Development Process

It has seven subcomponents, as illustrated in Fig. 1. The **AmI Applications Development** component regards the tools used for AmI applications development. **CAmIE Agent Model**, [3], extends the main lines of the agent models described by S-CLAIM, [1], and CLAIM, [4], to the problem of CAmIE. The **Compiler Back End** is modular, offering the possibility to implement any agent component with different technologies. The Analysis & Optimization subcomponent considers both agent class and AmI application level aspects, for optimizing the whole environment. The **Compiler Front End** is modular too and it can be extended for new languages and knowledge representation formalisms. The default *language* offered for agent class definition is CAmI, [3].

Translation and Interpretation consists of a repository of modules to translate between different knowledge representation formalisms and to adapt the content for interpretation by target ontologies. They are used by translation services or integrated in agents. **Visualization** offers an interface for the interaction of the users with the knowledge libraries, by representing and structuring their content in a user-friendly manner. The **Knowledge libraries** contain *general knowledge*, *best practices*, as tuples of the form (*context*, *goal*, *plan*) or (*context*, *plan*), where a plan is a list of associated agent behaviors, and a *mechanism of review*, for both the users and the developers, to assure the consistency of the content. A history of the evolution of the records is kept.

The *distribution platforms* encourage the developers contribute. They can contain: *AmI applications*, *knowledge libraries*, *languages*, *formalisms*, *ontologies* and *translation*, *interpretation*, *visualization* and *compiler* modules.

```
1  for each new agent, x_a, added to the multi-agent system
       A behind an executing AmI application {
2        for each related executing AmI application {
3              get the multi-agent system B behind it;
4              if there is a twin agent x_b ∈ B of x_a
5                    associate(x_a, x_b);
6        }
7  }
```

Fig. 3. Algorithm for connecting AmI applications, at runtime, inspired from [2]

2.4 The Execution Process

It has eight subcomponents, as seen in Fig. 2. **Multi-Agent Layer Management** keeps track of the AmI applications from the environment and of the connections made and possible. It assures consistency in the associations with real world entities too. **CAmIE Applications Execution** assures the deployment of agents and the execution of AmI applications, based on scenario description files. **Connection** facilitates the connection between AmI applications, based on the concept of *twin agents*, [3], using the algorithm from Fig. 3. **Composed Agents** gather the knowledge bases of more agents together and improve the context sensitivity, by opening the boundaries of the application-level agent hierarchies (see [3]). A composed agent can exist on more devices, based on good synchronization methods. **Translation and Interpretation** offers real-time translation and interpretation services, based on the modules mentioned in Section 2.3. Agents use them when they can't translate or interpret something on their own. Similarly, **Visualization** offers to the users real-time visualization services for the knowledge libraries. **Collective Knowledge Layer Management** indexes the published knowledge libraries and keeps track of important details, like how to access them. The **Knowledge Libraries** are used, based on

subscriptions, by agents and they are updated by agents or users. They can also be linked to other libraries and updated accordingly.

3 Conclusion and Perspectives

This work briefly describes the CAmIE model, for the bottom-up development of large-scale AmI environments, endowed with a high level of intelligence, using an agent-based approach. The validation of the CAmIE model is a work in progress. So far, the stand-alone AmI applications development was proven by the works on S-CLAIM, [1], and tATAmI, [6, 7], which are based on the same principles, on scenarios such as the SmartRoom and the Pro-Con case studies. Our future work comprises two more validation steps, one concerning the interactions between the AmI applications and the emergence of CI and the other concerning the ability of our model to create an entire ecosystem in order to reach the large-scale level envisioned. For this purpose, a platform for CAmIE based on tATAmI-2, [6], is currently being developed.

Acknowledgment

The work has been funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/132395.

References

1. Baljak, V., Benea, M.T., Seghrouchni, A.E.F., Herpson, C., Honiden, S., Nguyen, T.T.N., Olaru, A., Shimizu, R., Tei, K., Toriumi, S.: S-CLAIM: An agent-based programming language for AmI, a Smart-Room case study. Procedia Computer Science 10, 30–37 (2012)
2. Benea, M.T.: An agent-based solution for the problem of designing complex ambient intelligence systems. In: Intelligent Distributed Computing VII, Studies in Computational Intelligence, vol. 511, pp. 323–333. Springer (2014)
3. Benea, M.T., Florea, A.M., El Fallah Seghrouchni, A.: CAmI: An agent-oriented language for the collective development of AmI environments. In: Control Systems and Computer Science (CSCS), 2015 20th International Conference on. vol. 2, pp. 749–756. IEEE (May 2015)
4. El Fallah-Seghrouchni, A., Suna, A.: An unified framework for programming autonomous, intelligent and mobile agents. In: Multi-Agent Systems and Applications III, pp. 353–362. Springer (2003)
5. Howe, J.: Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business. Random House Digital, Inc. (2009)
6. Olaru, A.: tATAmI-2 – a Flexible Framework for Modular Agents. In: Control Systems and Computer Science (CSCS), 2015 20th International Conference on. pp. 703–710. IEEE (2015)
7. Olaru, A., Benea, M.T., Seghrouchni, A.E.F., Florea, A.M.: tATAmI: A platform for the development and deployment of agent-based AmI applications. Procedia Computer Science 52(0), 476 – 483 (2015), the ANT-2015 Conference