MODEL DRIVEN DEVELOPMENT OF AGENT BASED AMBIENT INTELLIGENCE PLACES WITH SERVICE ORIENTED ARCHITECTURE

Assist. Prof. Dr. Antonova I. D., Prof. Dr. Batchkova I. A., Eng. Ivanova Tz.,
Dept. of Industrial Automation, University of Chemical Technology and Metallurgy
Bul. Kl. Ohridski 8, Sofia, Bulgaria
i.antonova@uctm.edu, idilia@uctm.edu

Abstract: In recent years the use of computer technique and information technology in the domain of Ambient Intelligence (AmI) has been increasing significantly. For successful and efficient development of ambient intelligence systems and their components a new approach is needed. The Model Driven Development (MDD) is an innovative approach for development of different software applications using models at different levels of abstraction and applying model transformation to code generation. In order to satisfy the basic requirements to the developed AmI, in the proposed approach, the MDD, based on Model Driven Architecture (MDA), is directed to combine the use of agents as basic elements of the system and to model the internal and external communications in the system, based on the concept of Service Oriented Architecture (SOA). Such a way, the paper proposes a layered modeling framework for model driven development of agent based AmI systems with service oriented architecture. The deployment of the software on hardware resources is also part of the software process model. Finally some conclusions are made.

Keywords: AMBIENT INTELLIGENCE, MODEL DRIVEN DEVELOPMENT, MULTIAGENT SYSTEMS, SERVICE ORIENTED ARCHITECTURE, UML

1. Introduction

Market penetration of more and more diverse embedded and mobile devices set the agenda for solving the problem of ubiquitous computers dealing with that will make possible the successful implementation of diverse and heterogeneous applications in the area of Ambient Intelligence (AmI). The upcoming trends in the development of the software system in the domain of AmI are connected with the development and implementation of distributed information and control systems, consisting of multiplicity of nonhierarchical modules linked together via different types of communication systems. In order to control the complexity of such distributed real-time systems the following main requirements have to be fulfilled: reliable concept for decomposition and modularity, openness for extensions in the cases of new functionality services and devices, general architecture model of the system has to be producer independent, use of encapsulated, reusable components.

The response to these new challenges is to use new advanced methods in their development life cycle, such as the Model Driven Development (MDD) in order to meet the requirements for reusability, interoperability and cost efficiency. The second aspect is the architecture of the developed system. One very important task is to achieve modularity of the developed system and organizational interoperability between the different components based on Service Oriented Architecture (SOA). The third very important aspect in the system development is to achieve a reactive and proactive behavior of the system components through their modeling and implementation as agents, based on the fusion between SOA and Multi Agent Systems (MAS). The main aim of the proposed paper is to suggest an approach for development of distributed information and control systems for the domain of AmI, based on the combined use of above mentioned advanced approaches of software engineering.

The paper is organized in 5 parts. After the introduction, in part 2 a short analysis of research requirements for AmI is proposed. Part 3 discusses the main features of model driven development, multi-agent systems and service oriented architectures. In Part 4 of the paper the suggested approach is described. Finally some conclusions are made.

2. Short analysis of research requirements for achieving Ambient Intelligence space

Ambient Intelligence (AmI) is a vision of Information Society and Technology Advisory Group (ISTAG) [1] for building of smart environments that are reactive and proactive to people and make their actions safer, more efficient, more informed, more comfortable or simply more enticing. This means that the focus is moved from the individuals that has to adapted, towards of the technologies which need to be adapted to the individuals. As shown in Fig.1, AmI spaces are combinations of two kinds of components: components of operational technologies such as smart materials, I/O devices such as sensors an actuators, ubiquitous communications, adaptive software etc. and AI components including media management and handling, natural interaction, computational intelligence, context awareness, and emotional computing.

In recent years the use of computer technique and information technology in the domain of Ambient Intelligence (AmI) has been increasing significantly. For successful and efficient development of ambient intelligence systems and their components a new approach is needed. The Model Driven Development (MDD) is an innovative approach for development of different software applications using models at different levels of abstraction and applying model transformation to code generation. In order to satisfy the basic requirements to the developed AmI, in the proposed approach, the MDD, based on Model Driven Architecture (MDA), is directed to combine the use of agents as basic elements of the system and to model the internal and external communications in the system, based on the concept of Service Oriented Architecture (SOA). Such a way, the paper proposes a layered modeling framework for model driven development of agent based AmI systems with service oriented architecture. The deployment of the software on hardware resources is also part of the software process model. Finally some conclusions are made.

Keywords: AMBIENT INTELLIGENCE, MODEL DRIVEN DEVELOPMENT, MULTIAGENT SYSTEMS, SERVICE ORIENTED ARCHITECTURE, UML

1. Introduction

Market penetration of more and more diverse embedded and mobile devices set the agenda for solving the problem of ubiquitous computers dealing with that will make possible the successful implementation of diverse and heterogeneous applications in the area of Ambient Intelligence (AmI). The upcoming trends in the development of the software system in the domain of AmI are connected with the development and implementation of distributed information and control systems, consisting of multiplicity of nonhierarchical modules linked together via different types of communication systems. In order to control the complexity of such distributed real-time systems the following main requirements have to be fulfilled: reliable concept for decomposition and modularity, openness for extensions in the cases of new functionality services and devices, general architecture model of the system has to be producer independent, use of encapsulated, reusable components.

The response to these new challenges is to use new advanced methods in their development life cycle, such as the Model Driven Development (MDD) in order to meet the requirements for reusability, interoperability and cost efficiency. The second aspect is the architecture of the developed system. One very important task is to achieve modularity of the developed system and organizational interoperability between the different components based on Service Oriented Architecture (SOA). The third very important aspect in the system development is to achieve a reactive and proactive behavior of the system components through their modeling and implementation as agents, based on the fusion between SOA and Multi Agent Systems (MAS). The main aim of the proposed paper is to suggest an approach for development of distributed information and control systems for the domain of AmI, based on the combined use of above mentioned advanced approaches of software engineering.

The paper is organized in 5 parts. After the introduction, in part 2 a short analysis of research requirements for AmI is proposed. Part 3 discusses the main features of model driven development, multi-agent systems and service oriented architectures. In Part 4 of the paper the suggested approach is described. Finally some conclusions are made.

2. Short analysis of research requirements for achieving Ambient Intelligence space

Ambient Intelligence (AmI) is a vision of Information Society and Technology Advisory Group (ISTAG) [1] for building of smart environments that are reactive and proactive to people and make
These properties of AmI systems cannot be achieved without the use of new methods, technologies and tools, among which stand out the buzzwords pervasive, ubiquitous, context-aware, profiling etc. Another important factor in achieving the desired properties of the AmI system is the choice of suitable architecture and software infrastructure that enable easy integration, interoperability and extensibility. Considering these aspects, in this paper an infrastructure to support an efficient approach for development of AmI systems, combining the approaches of MDD, SOA and MAS, is presented. In the next part a short overview and comparison of these approaches is given.

3. Short review of the applied techniques

3.1. Model driven development (MDD)

Model Driven Engineering (MDE) and Model Driven Development (MDD) [3] are some of the most promising and challenging approaches for development and maintenance of highly distributed real time information and control systems such as the AmI systems. Here the systems are presented as models that conform to meta-models, and the model transformations are used to manipulate the various representations. The main difference from other development methods based on models is that MDD uses Models as inputs to parameterized implementation generators, i.e. implementation is (semi)automatically generated from the models.

Model Driven Architecture (MDA) [4] is a remarkable MDD initiative of Object Management Group (OMG), providing a powerful conceptual framework for development and transformation of three interconnected types of models - Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM) towards executable applications as shown in Fig.2. The main features of the models are:

- **CIM** – is also known as business or domain model that uses vocabulary to present the basic expectation from the system and to bridge the gap between domain experts and developers. This abstraction hides all specifications connected to the system implementation.
- **PIM** – is a view of the system without any details about implementation. Basic tasks of the PIM model are to model logical data, to establish dependencies and defining workflows and processes. Furthermore, PIM models must be sufficiently complete and accurate to ensure a higher degree of automated implementation of the models in the next layer (PSM).
- **PSM** – combine the PIM specifications with concrete platform information needed for enabling system execution, i.e. the basic role of this model is to ease the code generation using PIM and selected execution platform.

![Fig.2: MDA conceptual framework](image)

Traditional MDA infrastructure consists of a 4 layer hierarchy of models. The bottom layer is referred as M0 and holds the “user data”, i.e., the actual data objects the software is designed to manipulate. The next level is M1 and holds a “model” of the M0 user data. This is the level at which user models reside. Level M2 holds a “model” of the information at M1. Since it is a model of a (user) model, it is often referred to as a meta-model. Finally, level M3 is a mode of the information at M2, and is characterized as the meta-meta-model. For historical reasons it is also referred to as the MOF. Very important aspect of the MDA approach is the transformation between the different viewpoint models. Model transformation relies on a set of mapping rules between models that inform the transformer tool about the patterns it needs to follow to output the target model [5]. The MDA approach is supported by the OMG techniques like Unified Modeling Language (UML), Query View Transformation Specification (QVT) and Meta Object facility (MOF) [6]. In the core of MDA are the open standards, UML, MOF, XML, etc. UML [7] provide an integrated modeling framework, covering structural, functional and behavior descriptions. The UML notations support the development of various diagrams that reflect different aspects of the system in order to capture the full complexity in the phases of detailed analysis and system design.

MDD and MDA are suitable for use in the development of AmI systems and its combination with SOA or MAS, because of the opportunities they create through the applied meta-models and model transformations for providing reusability, verification and validation of models, integration and interoperability. There are some successfully developed agent meta-model and transformation means for development of MAS, such as for example: SODA, MACMAS, DML4M4AS, PIM4Agent, Pineapple, Sol, etc. A review of these tools and applications is presented in [8].

3.2. Service Oriented Architectures (SOA)

Service Oriented Architecture (SOA) is a software model in which the concept of a ‘service’ is an abstraction of a function used by an application and provides an architectural approach that brings the flexibility and agility required by today’s global business environment [9]. The model of Service Oriented Architecture (SOA) includes three main applications: application for providing a service, application for requiring a service and application for registration of service (Fig.3). The interactions between applications involve three basic operations: publish, find and bind. They are performed with the objects of web service model, which are: service and service description. Service requestor is a client application requesting data or functionality. To request data, the application must call the web service. Service provider contains the web service that can be used by any other application. Service registry contains a description of all services that the user creates. When request a web service, the service registry is looking for requested service and sends the search result to the service requestor.

![Fig.3: Model of SOA](image)

Different environments for development of Service oriented architectures and models are known. Some of the most popular and used are: W3C Web Service Architecture, OASIS SOA Reference model, ebXML, Semantic Web Services, JINI, OSGi etc. [10]. All these initiatives are collections of best practice principles and patterns in service-oriented design. The W3C Web Services Architecture as one of the most promising architectures identifies
the functional components of Web Service architecture and defines the relationships among those components in order to affect the desired properties of the overall architecture [11]. Web Services generally use either SOAP or XML-RPC as communication method for services, UDDI for service discovery and WSDL for service description. Web services are process driven and use process model to describe process orchestration. Orchestration is a composition of business objects in process flows through defining the interactions between them including business logic and execution order. Choreographies on the other hand define collaborations between interaction parties.

MDD methodology may be used as a starting point for development of SOA based applications. For example, Rational Software has provided an extension to Rational Unified Process (RUP) called RUP-SOMA, built by IBM for service-oriented analysis and design and used for identification, specification, and realization of services, service components, and flows [12]. Another contribution towards the integration of MDD and SOA is the UML profile for modeling services and SOA named SoaML profile (http://www.SoaML.org, http://www.omg.org/spec/SoaML/), which is proposed by OMG, and allows expanding the capabilities of UML tools in the direction of modeling the basic elements and constructs of SOA. The joint use of SOA and MDA ("ModelPro" engine for Eclipse) reduces the development time of SOA applications, facilitating their maintenance and prolongs their lifespan.

### 3.3. Multi Agent Systems (MAS)

Multi Agent Systems (MAS) can be defined as "a loosely coupled network of problem solvers (agents) that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver" [13]. Jennings and Wooldridge [14] have defined an agent as "a computer system situated in some environment and capable of autonomous action in this environment, in order to meet its design objectives". Agents have the following main properties and characteristics [15]:

- **autonomy**: agents encapsulate some state (that is not accessible to other agents), and make decisions about what to do, based on this state, without the direct intervention of humans or others;
- **socialability (interactivity)**: agents interact with other agents (and possibly humans) via some kind of agent-communication language, and typically have the ability to engage in social activities (such as cooperative problem solving or negotiation) in order to achieve their goals;
- **reactivity**: agents are situated in an environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, or perhaps many of these combined), are able to perceive this environment (through the use of potentially imperfect sensors), and are able to respond in a timely fashion to changes that occur in it;
- **pro-activeness**: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative;
- **mobility**: agents can transport themselves across different systems architectures and platforms.

The agent community has considerable interest in developing methods and techniques for specifying, modelling, implementing and verifying of MAS for distributed information and control systems, as are the Aml systems, but so far no standardized design methodology has been recognized. Several object-oriented methodologies have been suggested for agent-oriented analysis and design, based on UML. Important drawbacks of using UML to model MAS are the modelling of agent communications as method invocations and the absence of references to the mental state of the agents. To overcome these drawbacks, the UML notations are extended to reflect the characteristic properties of the agents. Successfully extensions of UML are achieved in AUML, GAIA, MESSAGE/UML, AgentUML, Prometheus, etc. Some of them are based on FIPA standard (http://www.fipa.org) suggesting an agent reference model for creation, registration, location, communication, migration and retirement of agents. Recently are also available some specialized tools for lightweight devices, some of them suitable for the domain of Aml, such as DSML4MAS (http://dsml4mas.sourceforge.net/), FIPA-OS, ASEME (for Eclipse), Tropos (http://www.troposproject.org/), INGENIAS (http://sourceforge.net/projects/ingenias/), Jade-Leap, etc. However, there are limitations and drawbacks, associated with the variety of devices and communication protocols, specific for Aml. As well there are some agent-based development environments especially for the Aml domain, as for example: THOMAS, MaRV, ALZ-

The discussion around the fusion of MAS and MDD methodology may be used as a starting point for development of SOA based applications. For example, Rational Software has provided an extension to Rational Unified Process (RUP) called RUP-SOMA, built by IBM for service-oriented analysis and design and used for identification, specification, and realization of services, service components, and flows [12]. Another contribution towards the integration of MDD and SOA is the UML profile for modeling services and SOA named SoaML profile (http://www.SoaML.org, http://www.omg.org/spec/SoaML/), which is proposed by OMG, and allows expanding the capabilities of UML tools in the direction of modeling the basic elements and constructs of SOA. The joint use of SOA and MDA (“ModelPro” engine for Eclipse) reduces the development time of SOA applications, facilitating their maintenance and prolongs their lifespan.

#### 4. Description of the suggested approach

In order to fulfill the basic requirements to the Aml systems, the software infrastructure of the system is necessary to be distributed, enabling self-organization of devices and their software components. Aml systems are much more complicated than traditional computing systems. Hence, characteristics such as adaptability, flexibility, interoperability and modularity are more important. Furthermore, these systems must provide common improvements such as service discovery, self-organization, rich knowledge representations and context-awareness.

The suggested model driven approach supporting the fulfillment of above mentioned requirements uses MDA and comprises the whole development life cycle of software development, starting with CIM meta-model to the deployment. The approach is illustrated in Fig.4 using 4 layers: one CIM layer, two PIM layers and one PSM layer. Four principles as architecture cornerstones are chosen:

- **Model-driven principle** using MDA of OMG, supporting the platform independent and model based domain engineering;
- **Service-oriented architecture using PIM4SOA meta-model** (http://pim4soa.sourceforge.net/);
- **Web services meta-model for description of loosely coupled, reusable, composable components**;
- **PIM4Agents meta-model** - platform-independent model that specifies multiagent systems in a technology independent manner.

With the proposed approach it is possible to design an agent-based Aml system with service oriented architecture specifying high level concepts in a platform-independent agent model (focusing on the domain model), and later automatically transform it for different implementation models, bridging the gap between design and implementation. The use of two layers of PIM models enables for horizontal transformation between the models of the second layer and the use of “foreign” environments for performance.

The platform-independent model for service-oriented architectures (PIM4SOA) covers four important aspects:
information, services, processes and quality of service. For example the services aspect include the services, which are represented as collaborations specifying pattern of interactions between participating roles, as shown in Fig.5.

Fig.4: Illustration of the suggested approach

Fig.5: Subset of PIM4SOA for service aspect

PIM4Agents meta-model represents an integrated view on agents in which different components can be deployed on different execution platforms. The PIM4Agents meta-model defines modeling concepts that can be used to model different aspects or views of an agent system, such as: Agent, Organization, Role, Interaction, Behavioral, Environment etc., as is shown in Fig.6.

Fig.6: Basic Views of PIM4Agents meta-model

5. Conclusions

The suggested approach uses the MDA to address the modeling of MAS and SOA in order to facilitate software development through abstractions avoiding the specific implementation details. The proposed meta-models are implemented in EMF of Eclipse and are presented in ECORE format. The transformation PIM and PSM may be done by using ATLAS transformation language (ATL). The main idea is to achieve a layer based design that enables the division of system elements into levels, reducing the coupling between modules, facilitating abstraction as well as the distribution of responsibilities. Combining SOA with MAS will improve the characteristics of the system being developed in the direction of meeting the requirements of anticipatory and adaptability. The choice of PIM meta-models is essential to achieve the requirement for context awareness of AmI systems.

6. References

5. Gorp P., Model-Driven Development of Model Transformations