LNCS 7852

Jörg P. Müller
Massimo Cossentino (Eds.)

Agent-Oriented Software Engineering XIII

13th International Workshop, AOSE 2012
Valencia, Spain, June 2012
Revised Selected Papers

Springer

Jörg P. Müller   Massimo Cossentino (Eds.)

Agent-Oriented Software Engineering XIII

13th International Workshop, AOSE 2012
Valencia, Spain, June 4, 2012
Revised Selected Papers

Springer

Volume Editors

Jörg P. Müller
Technische Universität Clausthal, Institut für Informatik
38678 Clausthal-Zellerfeld, Germany
E-mail: joerg.mueller@tu-clausthal.de

Massimo Cossentino
Istituto di Calcolo e Reti ad Alte Prestazioni, Consiglio Nazionale delle Ricerche
90128 Palermo, Italy
E-mail: cossentino@pa.icar.cnr.it

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-39865-0 e-ISBN 978-3-642-39866-7
DOI 10.1007/978-3-642-39866-7
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013943815

CR Subject Classification (1998): I.2.11, D.2, I.2, D.1, D.3, I.6

LNCS Sublibrary: SL 2 – Programming and Software Engineering

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Since the mid-1980s, software agents and multiagent systems have grown into a very active area of research and of commercial development activity. One of the limiting factors in the industry take-up of agent technology, however, is the lack of adequate software engineering support and knowledge in this area. The Agent-Oriented Software Engineering (AOSE) Workshop is focused on this problem and provides a forum for those who study the synergies between software engineering and agent research. The concept of an agent as an autonomous system, capable of interacting with other agents in order to satisfy its design objectives, is a natural one for software designers. Just as we can understand many systems as being composed of essentially passive objects, which have state, and upon which we can perform operations, so we can understand many others as being made up of interacting, autonomous or semi-autonomous agents. This paradigm is especially suited to complex systems. Software architectures that contain many dynamically interacting components, each with their own thread of control, and engaging in complex coordination protocols, are typically orders of magnitude more complex to correctly and efficiently engineer than those that simply compute a function of some input through a single thread of control, or through a limited set of strictly synchronized threads of control. Agent-oriented modeling techniques are especially useful in such applications.

The 12 past editions of the agent-oriented software engineering workshop (AOSE) had a key role in this endeavor. For the 13th AOSE workshop held during the 11th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), the thematic focus was on exploring the new emerging role of agent-oriented software engineering as a bridge from the now consolidated agent-oriented programming languages and platforms, to recent systems modeling paradigms such as self-*, autonomic systems, and systems of systems (SoS). Thus, the theme of this workshop was to explore, from an agent-based perspective, foundations, models, methods, architectures, and tools for engineering future software-intensive IT ecosystems.

The AOSE 2012 workshop received 24 submissions. Each paper was peer-reviewed by three members of an international Program Committee. The papers in this volume include both selected and thoroughly revised papers from the AOSE 2012 workshop and two invited papers. The papers cover a broad range of topics related to software engineering of agent-based systems, with particular attention to integration of concepts and techniques from multiagent systems with recent programming languages, platforms, and established software engineering methodologies. We hope that this volume will stimulate further research in agent-oriented software engineering as well as its integration with conventional software engineering.

This volume is special in another respect, too: It documents the results of what is very likely to have been the last AOSE workshop. The reason for this is that from 2013 onwards, AOSE will be merging with two other notable events, the International Workshop on Programming Multi-Agent Systems (ProMAS) and the International Workshop on Declarative Agents Languages and Technologies (DALT), to form a new event, the International Workshop on Engineering Multiagent Systems (EMAS). The first edition of EMAS will be held at the AAMAS 2013 conference. It is our hope that the merger of the three major scientific workshops on software engineering for multiagent systems will sustainably strengthen our research field and create new impact for research directed toward engineering large-scale, distributed software systems.

We wish to express our gratefulness to the AAMAS 2012 organizers for hosting AOSE. We thank the AOSE PC members and auxiliary reviewers for their thorough, critical, and constructive review work. We are grateful to the AOSE Steering Committee for their continued support and advice. Finally, we thank the Springer staff headed by Alfred Hofmann for accompanying the AOSE workshop over the past 13 years and for supporting the publication of this volume.

May 2013

Jörg P. Müller
Massimo Cossentino

Organization

The AOSE 2012 workshop was organized in colocation with the 11th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS) which was held in Valencia, Spain in June 2012.

AOSE 2012 Chairs

Jörg P. Müller	TU Clausthal, Germany
Massimo Cossentino	National Research Council of Italy, Italy

Program Committee

Carole Bernon
Olivier Boissier
Juan Antonio Botia Blaya
Lars Braubach
Scott Deloach
Amal El Fallah Seghrouchni
Maksims Fiosins
Klaus Fischer
Giancarlo Fortino
Ruben Fuentes-Fernández
Aditya Ghose
Holger Giese
Paolo Giorgini
Adriana Giret
Marie-Pierre Gleizes
Alma Gomez-Rodriguez
Jorge J. Gómez Sanz
Vincent Hilaire
Lam-Son Lê
Joao Leite
João G. Martins
Philippe Mathieu
Frédéric Migeon
Ambra Molesini
Pavlos Moraitis
Juan Carlos Gonzalez Moreno
Haralambos Mouratidis
Andrea Omicini
Flavio Oquendo
H. Van Dyke Parunak
Juan Pavón
Michal Pěchouček
Gauthier Picard
Alexander Pokahr
Alessandro Ricci
Fariba Sadri
Valeria Seidita
Onn Shehory
Carles Sierra
Nikolaos Spanoudakis
Angelo Susi
Kuldar Taveter
László Zsolt Varga
Danny Weyns
Eric Yu

AOSE Steering Committee

Paolo Giorgini	University of Trento, Italy
Jörg P. Müller	TU Clausthal, Germany

Gerhard Weiss	Maastricht University, The Netherlands
Danny Weyns	Linnaeus University, Sweden
Michael Winikoff	University of Otago, New Zealand

Table of Contents

A Methodological Approach to Model Driven Design of Multiagent Systems

Klaus Fischer and Stefan Warwas

German Research Center for Artificial Intelligence (DFKI) GmbH
Campus D3_2, 66123 Saarbrücken, Germany
`firstname.lastname@dfki.de`

Abstract. In this paper we propose a methodological approach to model driven design of multiagent systems (MAS). However, several methodologies for MAS have already been proposed and we do not want to present yet another new methodology. Our aim is rather to explain how our MAS development framework Bochica, which we already presented in [1], relates to such methodologies and how the proposals from literature can be integrated to extend the Bochica framework. As a result, we propose an iterative process for MAS design where several stakeholders work cooperatively in a food chain for the design of MAS and each stakeholder gets the tool support that he or she needs.

1 Introduction

The multiagent system (MAS) research group at the German Research Center for Artificial Intelligence (DFKI) GmbH has a long history in research on MAS design as well as in the development of MAS that are in practical use in an industrial setting. The most complex of these systems is a shop floor control system that is in daily use 24/7 in the steel work of Saarstahl AG in Völklingen. All the experience from this work went into the design and implementation of the Bochica framework for the model driven design of MAS. Bochica aims in the first place at the system engineer who wants to adopt an agent-based approach to a specific problem in some application domain. The major use of Bochica in the MAS research group is currently in the area of MAS design for agents that solve problems in virtual 3D environments.

At least in our work so far we did not put much effort into the investigation or use of methodologies in our system development but consider this fact as a major drawback of our contribution. However, Bochica is well-suited to be embedded into existing methodologies and provides explicit support for such an endeavor. In the first place a model driven approach works best if a system can be designed and developed in a top down manner. However, if at all, such an approach is only applicable in situations where a system is developed from scratch and such situations are in practice rather an exception than the regular case. Most of the time the system under considerations (SUC) is already designed and/or implemented partially or already existing subsystems have to be included into a fresh system architecture and design. Bochica's aim is to support round-trip

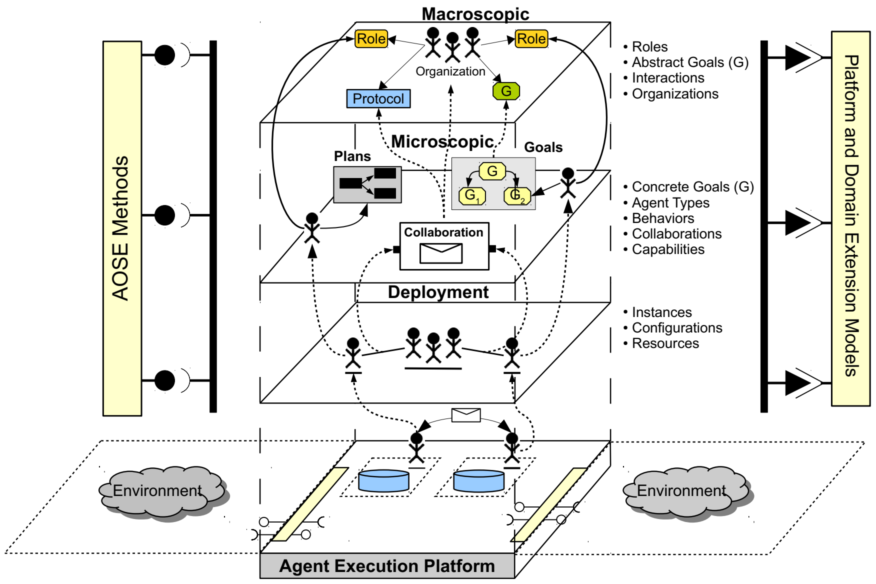


Fig. 1. The BOCHICA Framework

engineering by supporting reverse engineering of models form implementations. Therefore, a methodology that supports iterative and agile system development is best suited to extend the BOCHICA framework.

Furthermore, usually a group of engineers is involved in the design of complex systems and therefore collaborative system design and modeling is an issue that we want to deal with in the further development of BOCHICA. Views and viewpoints are important concepts to support different stakeholders in the development process and to give them access exactly to the information they need.

2 The BOCHICA Framework

In this section we briefly summarize the BOCHICA framework for model driven AOSE. It has been initially introduced in [1]. BOCHICA (see Figure 1) evolved from research on a platform independent metamodel and tool chain for the design of MAS. The role of BOCHICA in the overall software development process is to provide the means for capturing design decisions and bridging the gap between design and code. This raises the question of how BOCHICA can be integrated with typical software development processes. As of today, iterative and agile development processes turned out to be more appropriate for most software projects than sequential ones. Most agent-oriented methodologies also propose to use an iterative development processes (see Section 3). As depicted in Figure 1, the core domain specific language (DSL) underlying the Bochica framework is structured into three layers. We distinguish between the macroscopic, microscopic, and deployment layer:

Macroscopic. The overall structure of a MAS is specified by Dsml4Mas[1] (cf. [2]) in terms of organizational structures. The responsibilities inside an `Organization` are represented by `DomainRoles` and `AbstractGoals`. The communication between the involved parties is defined by `Interactions`. An `Interaction` defines the valid message sequences. The design artifacts of the macroscopic layer serve as a contract between the agents. They can be used for deriving the basic structure of the microscopic layer. However, how every agent fulfills the requirements is left open to the agent.

Microscopic. The microscopic layer of Bochica defines concepts for modeling the internals of agents. This encompasses concepts like `Behavior`, `Event`, `Resource`, `KnowledgeBase`, and `Collaboration`. A `Behavior` specifies the behavior of agents by `Activities` which are connected by `Flows`. `ConcreteGoals` are used to refine the `AbstractGoals` from the macroscopic layer. The concept of `Expression` is used to abstract from concrete software languages. For example, the software language for defining a `BooleanExpression` always depends on the concrete scenario. Bochica abstracts from those details and provides extension interfaces for plugging in 3^{rd}-party languages. Likewise, the `KnowledgeBase` concept abstracts from concrete knowledge representation languages. The internals of `Organizations` are defined by `Collaborations`. A `Collaboration` exactly specifies the bindings between roles of an `Organization` and `Actors` of an `Interaction`. Moreover, `ProtocolConfigurations` are used to instantiate the abstract `Interaction` of the macroscopic layer with concrete content types, time out values, and role bindings.

Deployment. The deployment layer specifies concrete instances of agents and organizations defined by the microscopic and macroscopic layers. This includes the initialization of concrete role fillers of organizational roles. Moreover, an `AgentInstance` contains `Initializers` for specifying the initial beliefs and goals.

In the following we summarize the new features Bochica with respect to what already was present in Dsml4Mas [2]:

Expressiveness. Expressive modeling languages are required for closing the gap between models and code. For this purpose, we further developed the underlying core modeling language so that large portions of the source code can be generated.

Conceptual Extensions. The Bochica framework offers various interface concepts that can be extended through external plug-ins. For example, existing concepts can be specialized for certain application domains or execution environments. Moreover, new ways for modeling existing aspects can be contributed (e.g. behaviors or interactions).

[1] Domain Specific Modelling Language for MAS.

Language Extensions. A large number of software languages are around that are relevant for developing agent-based systems such as knowledge representation languages, query languages, or programming languages. Bochica provides abstract language interfaces such as `BooleanExpression` or `ContextCondition` which can be extended by external language plug-ins. The interfaces check syntactical correctness and the binding of variable symbols in the surrounding scope.

Transformations. The Bochica framework uses so called *base transformations* for mapping the concepts of the core DSL to concepts of the target execution environment. As Bochica gets extended with new concepts, a so called *extension transformation* extends the base transformation for the new concepts. Currently, we have a base transformation for Jadex which is implemented in QVT.

Reusability. It is desirable to reuse model artifacts that proved their practical use and were validated (e.g. interaction protocols or goal hierarchies). For this purpose, we established a reverse engineering approach for extracting the underlying structure of Jadex BDI agents [3]. The approach is used to build up model repositories and ease the migration of existing projects to Bochica.

3 Related Work on Agent-Oriented Design Methodologies

Several software development processes like the classical *waterfall model* [4] and the iterative *spiral model* [5] originated from traditional software engineering. During the recent years, iterative and agile development processes gained more and more attention by software developers. For example, the *Rational Unified Process* (RUP) [6] is a widely accepted iterative development process and provides a customizable framework for configuring the development process. RUP uses UML for capturing the design decisions. According to [6], RUP distinguishes between the four phases *Inception*, *Elaboration*, *Construction*, and *Transition*. RUP follows the idea of producing a prototype of the system in each iteration. This means that each phase undergoes at least once the whole iteration cycle from requirements to code and produces a deployable artifact. Each phase in RUP is dedicated to answers different questions. For example, the Inception phase focuses on determining the feasibility of the overall project, while later iterations phases narrow down the concrete software architecture. Thus, the possibility to produce early prototypes which can be refined in later iterations is important for RUP. A further output of the Inception phase is to define the concrete development process (e.g. the utilized methods) and the used tools.

It has been widely recognized within the agent community that the existing software engineering methodologies do not satisfy the needs of AOSE (e.g. [7], [8, p. 22]). During the recent years, various agent-oriented methodologies have been proposed. The *FIPA Methodology Technical Committee*[2] and the *FIPA Working Group: Design Process Documentation and Fragmentation*[3] are two initiatives for the unification and standardization of agent-oriented methodologies. As of

[2] `http://www.fipa.org/activities/methodology.html`
[3] `http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/`

today, there exists no standardized agent-oriented approach and the methodologies are still driven by research. The Bochica framework is related to agent methodologies as it provides the means for capturing design decisions and bridging the gap between high-level designs and executable code. In the following we give a brief overview of the agent-oriented design methodologies which we consider most important for our approach. For the lack of space and time it is not possible to give a complete overview over the state of the art in this section. [9], [10], [11], and [12] provide a good overview of the state-of-the-art. The methodologies were selected due to their influence in the community and the relevance to our approach.

Gaia ([13] and [7]) is an agent-oriented methodology which follows a sequential development process. Gaia covers the agent-oriented analysis and design phases. The design artifacts are kept abstract and leave many aspects open (e.g. concrete interaction protocols or behavior patterns are not defined). Gaia highlights the role of organizational structures and the environment. During the analysis phase, organizational structures, interactions, and an environment model are defined. The architectural and detailed design phases further refine the models by adding agent and service models.

INGENIAS ([14], [15], and [16]) is an agent-oriented methodology which supports the development of agents with a mental model. INGENIAS originated from the MESSAGE [17] methodology and is aligned to RUP. Much research effort has been spent on the detailed design and implementation. In [14], testing and debugging of interaction protocols in INGENIAS was discussed. In order to unify the benefits of INGENIAS with other approaches, the combination with Tropos [18] and Prometheus [19], was discussed [20].

The **O-MaSE**[4] ([21], [22] and [23]) methodology has an organizational view on AOSE. For example, it supports *policies* for constraining the behavior of a system. The O-MaSE methodology does not define a fixed development processes. Instead, O-MaSE provides a framework for combining different *method fragments* for the requirements, analysis, and design phases. *Method construction guidelines* support this process. The process framework was initially based on the *Open Process Framework* (OPF) [24] and was migrated to SPEM. According to [23], O-MaSE has been evaluated in sequential and iterative development processes.

Prometheus[5] ([8]) is a methodology for developing BDI agent systems. It covers the three development phases (i) system specification, (ii) architectural design, and (iii) detailed design. Testing and debugging has been discussed by [25], [26], and [27]. During the system specification phase, system goals, typical processes of a system (called *scenarios*), and perceptions and actions are collected. Similar goals, perceptions, and actions are grouped to *functionalities*. The architectural and detailed design phases are concerned with identifying agent and capability types by grouping functionalities and specifying interaction protocols

[4] Organization-based Multiagent System Engineering.

[5] `http://www.cs.rmit.edu.au/agents/SAC2/methodology.html`

using AUML. The integration of AUML for specifying interaction protocols has been discussed by [28] and [29].

PASSI[6] ([30]) comprehends the construction of five models (System Requirements, Agent Society, Agent Implementation, Code Model, and Deployment Model). The developers behind PASSI provide an add-in for the commercial UML-based CASE tool Rational Rose which allows to combine the PASSI development process with UML-based modelling for the design of the system.

Tropos[7] ([31], [32] and [33]) is an agent-oriented methodology that covers the development phases reaching form early and late requirements to architectural and detailed design. In Tropos, models undergo an incremental step-wise refinement. The development process starts with identifying actors (stakeholders), the system's goals and their dependencies. Goals are further decomposed into sub-goals and means-end-analysis is used for identifying plans and resources (means) necessary for achieving a goal (end). The architectural design phase is concerned with identifying sub-actors and specifying information and control flows. The detailed design phase uses UML activity diagrams for defining the behavior of agents and AUML sequence diagrams for the interaction between agents. It has been discussed how Tropos can be used for developing Jack [32], Jade [33], and Jadex [34] agent systems. There exists also an extension of Tropos for adaptive systems [35].

There are a number of other approaches like for example ASPECS [36], MASSIVE [37], MOISE+ [38], and OMNI [39] which make valuable contributions regarding the design of MAS in different directions.

The different approaches listed above have two commonalities: (i) most of them are dedicated to BDI agents and (ii) the majority proposes to follow an iterative development process. The methodologies differ in their foci – e.g. on early are late phases. Currently, the agent-oriented methodologies are undergoing a consolidation phase. There are several initiatives for unifying the different approaches using process languages like SPEM. We expect that this phase will continue for some more time.

Our aim is not to define yet another methodology but to integrate existing methodologies in our framework. The selection of concrete methods and the definition of the development process is orthogonal to the Bochica framework which we describe in more detail. Bochica provides the concept `MethodologyArtifact` as interface to such methodologies. Instead of having a separate modeling language and tool for each methodology, most of the methodologies could be integrated into one framework and share a common core. This would join the efforts of the involved parties and would ease the maintenance of the tool chain. In case the methodology requires additional model artifacts for capturing the design decisions which can be integrated into Bochica by extending the provided interfaces. With this Bochica complements the FIPA activity *FIPA Design Process Documentation and Fragmentation Working Group* (cf. [40]) which tries to standardize the development and documentation process of MAS design.

[6] Process for Agent Societies Specification and Implementation.

[7] `http://troposproject.org`

The database of model fragments which is one of the aims of this activity could be linked to methodology plugins provided for Bochica. In this sense the Bochica framework can act as a mediator between standardized development processes for MAS and target execution platforms which are supported by the Bochica framework.

4 Extending Bochica

In the following, we take the iterative RUP methodology as basis for the discussion of how Bochica can be aligned to a typical software development process. We assume there exists a plug-in repositories for Bochica which enables the reuse of domain, methodology, and execution environment specific extensions across software development projects. Plug-in vendors maintain and distribute those plug-ins. At the beginning of each software project, the Bochica framework is configured with a basic set of plug-ins. If the requirements regarding the framework and the plug-ins are not clear, vanilla Bochica can be used for a basic evaluation. If there are no plug-ins available which cover the required functionality or the plug-ins only cover it partially, new extensions have to be developed or existing ones have to be customized. In most cases some customization will be necessary. For this purpose, we propose an iterative extension mechanism which is being discussed in the following.

We can distinguish at least two development cycles. On the one hand, it is the cycle of the tool development, on the other hand, the development cycle of the SUC for some application domain. It is clear that in the positive case there will be a significant number of instances for the development of systems for different application domain and that these cycles will have a higher frequency what the number of iteration concerns than the cycle for tool development. The feedback between the two cycles is given on the one hand by requirements and feature request from the application engineers and domain experts and on the other hand by new versions of the modeling tool chain by the tool developers.

In the following, Section 4.1 explains how Bochica itself should be further developed according to RUP, Section 4.2 defines the stakeholders in MAS design and their tasks for the application of Bochica, Section 4.3 discusses the role collaboration in AOSE, and Section 4.4 introduces the concept of viewpoints to support different stakeholders.

4.1 Evolution Process of Bochica

Figure 2 b) depicts an iterative rapid prototyping process for Bochica which is aligned to RUP. In the first iteration, the core concepts of the framework and model transformations are used to develop an early prototype of the system. Different execution platforms are evaluated by reusing existing *base transformations*. Based on the gained experiences, the requirements regarding the framework are collected. Later iterations, use the requirements for extending the Bochica metamodel with new concepts and the base transformation with

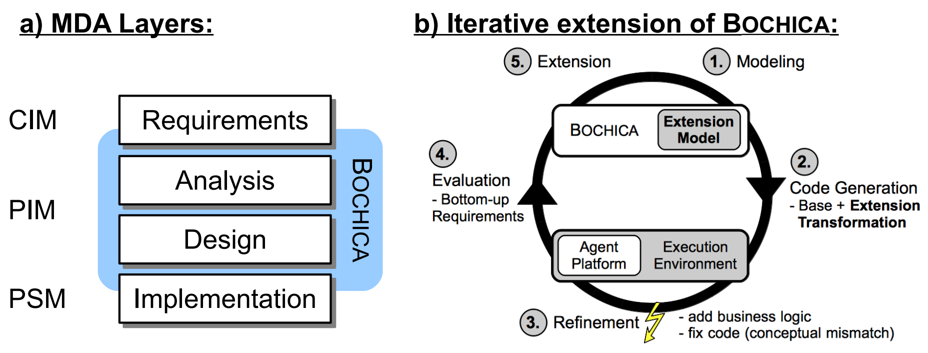


Fig. 2. (a) depicts the different abstraction layers as defined by model driven architecture (MDA) [41]. The blue box indicates that the conceptual framework underlying Bochica is dedicated to the analysis and design layers. The ability to extend Bochica with additional method, application domain, and platform specific concepts simplifies the transition from computational independent models (CIM) to platform independent models (PIM) and from PIM to platform specific models (PSM). (b) An iterative process for the extension of the Bochica framework.

an *extension transformation*. The extensions include custom concepts for the application domain, used methods, and the execution environment.The steps in the evolution of Bochica can be characterized as follows:

Modeling. The Bochica core DSL is used to capture the design decisions of the SUC. As the core DSL gets extended by an extension model in later iterations, the framework minimizes the need for customizations at code level.

Code Generation. A forward transformation is used to automatically generate code for a target environment. We distinguish between a *base transformation* which maps the concepts of the core DSL to code and an *extension transformation* which complements a base transformation with mapping rules for the custom concepts of the extension model.

Refinement. The generated code is refined by adding business logic where necessary. Concepts for capturing the design decisions for the SUC which are not covered by the modeling language and/or the model transformation are manually refined at the code level.

Evaluation. Aspects of the SUC which required manual refinements of the generated code are candidates for further extensions of Bochica. This is especially the case when there are no adequate concepts for expressing important design decisions. We call those requirements *bottom-up requirements.*

Extension. The collected requirements are used to create an extension model that extends the Bochica core DSL with missing concepts. Moreover, required views and tools are integrated. Finally, an extension transformations is created with additional mapping rules for the custom concepts.

4.2 Stakeholders in MAS Design

The application of the Bochica framework requires the interplay of different stakeholders. Stakeholders take different roles in the development process where each role can have a possibly large number of role fillers depending on the complexity of the SUC. In the following we characterize the involved parties and define their tasks. We separate the different stakeholders regarding the two development cycles. We do not consider our list of stakeholders to be complete. The stakeholders that we discuss here are meant to be illustrations of what we have in mind. Regarding the development cycle of tool development we distinguish the following stakeholders:

Methodology Expert: Integration of design methodologies derived from research on agent system design and general software engineering.

Language Engineer: Since Bochica is based on a DSL, the language engineer is responsible for extending it with new concepts. Detailed knowledge of the core DSL and the underlying metamodel is required to align new concepts to existing ones. We distinguish between (a) language engineers who further develop the core DSL and (b) those who create 3rd-party plug-ins. The firmer are not involved in the development process as depicted in the previous section. The latter are involved in the evaluation, requirements, and extension tasks. The language engineer has to choose the right level of abstraction for the conceptual extensions.

Tool Developer: The tool developer is responsible for building the development environment based on the DSL. This includes (i) writing model transformations (ii) creating new or extended diagrams, and (iii) providing further usability extensions such as wizards and additional tools. He has to make sure that the tools cover the (required) functionality of the target platform. The tool developer is involved in the extension task.

Regarding system development for application domains we consider the following stakeholders (see also Figure 3):

Agent Engineer: The agent engineer is the end user of the development environment. According to his needs, he installs the required plug-ins and uses an agent methodology to design the MAS. Model repositories are used to cooperate with colleagues and reuses existing model artifacts. The agent engineer is also responsible to refine the generated code where necessary. He is involved in the modeling, code generation, and evaluation tasks.

Protocol Experts: Protocol experts are responsible for specifying communication and negotiation protocols. Research results (e.g. about the properties of a specific auction protocol) can be directly linked to model artifacts in the model repository. This information helps engineers in constructing MAS.

BDI Expert: Example for theory experts who know a specific theory (in this case BDI) very well and can apply it in the system development for the application domain.

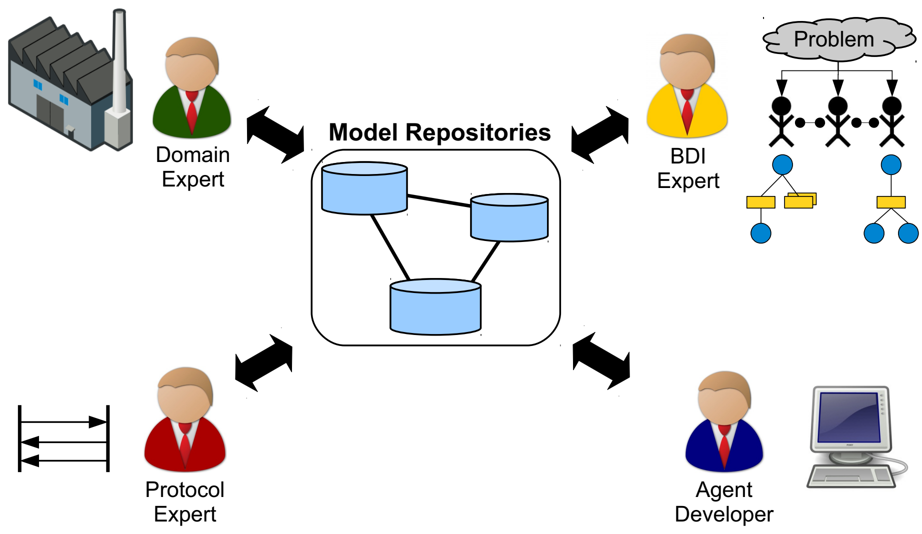


Fig. 3. This figure depicts typical stakeholders involved in the development of a MAS for some application domain. Each stakeholder uses specialized viewpoints to create model artifacts which are shared and reused through model repositories. A similar diagram can be given for the tool development where of course a different set of stakeholders is involved and the repository these stakeholders are working on is rather a plug-in repository.

Domain Expert: Domain experts have the knowledge regarding the application domain for the SUC.

Regarding the use of the MAS design methodologies listed in Section 3 different approaches are possible. One can use any of the listed methodologies to produce a separate design and manually translate it into model artifacts of Bochica. Such an approach is feasible but not really desirable because in the manual translation loss of information and missinterpretation is possible. The better way to include a concrete methodology is to provide a plugin which directly supports this methodology. If additional concepts are needed which are not yet available in Bochica, an extension of Bochica's metamodel has to be provided for this methodology. On top of the extended metamodel the plugin can be implemented which would then provide tool support for the selected design methodology. Because the artifacts which are produced in the development process when the methodology is applied would be directly part of the model instance which is created, only the model transformations which translate the model instance into executable code would need to be extended.

With the resources which we can spend on the development of Bochica, it is impossible to provide a metamodel extension and a plugin for each methodology which is presented in literature. For this we limit our contribution to provide the framework. The research community will decide on whether our proposal is

Fig. 4. Relationships between Patterns, Aspects, and Viewpoints (from [42], p. 6)

beneficial and whether the methodology designers see the benefits of the Bochica framework to go through the trouble of providing the metamodel extensions and the respective plugins.

4.3 Collaboration

The development of large scale software systems requires the collaboration of many different stakeholders. Each stakeholder requires specialized views which enable him to abstract form other aspects. Validated and tested model artifacts are shared through model repositories. Design patterns can be reused as blue print for similar scenarios and execution platforms. Methods for model-driven reverse engineering enable the reuse of design patterns of already implemented MAS. Currently, we are using a file-based approach for sharing model artifacts but native model repositories are already becoming available (e.g. CDO[8]).

4.4 Views and Viewpoints

Bochica's modeling tool supports views that allow the system engineer to view fragments of the underlying system in a diagram in a graphical manner. For different stakeholders it is not desirable to give all of them access to the full set of available views. The collection of views for a specific stakeholder makes up a viewpoint. The most easy way to deal with access restrictions regarding viewpoints is to provide a separate tool for each viewpoint. In this case the operating system is dealing with all problems of security, at least what pure access to the viewpoints is concerned. If security aspects in the interaction of the viewpoints is an issue, of course, additional mechanisms are necessary but such mechanisms are out of scope for the discussion in this paper.

Figure 4 shows how the concept viewpoint relates to other concepts that are important for system design. According to this diagram a viewpoint is dedicated

[8] `http://www.eclipse.org/cdo/`

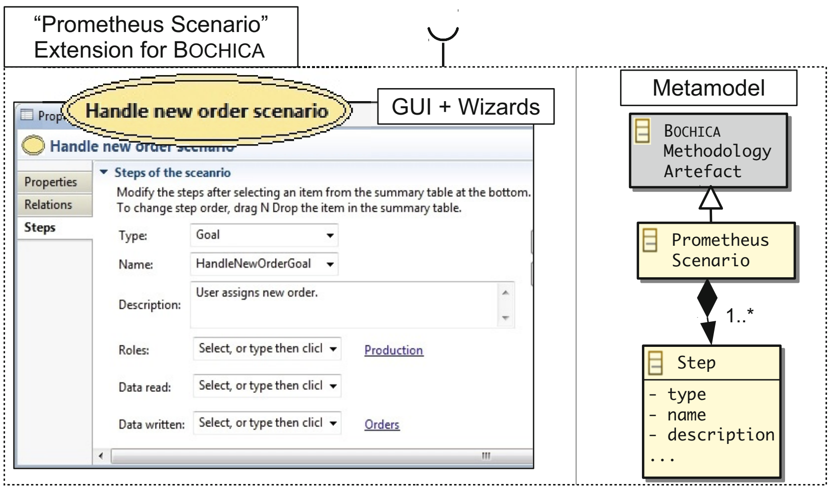
to concerns that should be handled by a specific stakeholder and by using views this stakeholder can access the model instance that corresponds to a system where several model instances can contribute to this system. Most important to notice is that the viewpoint is basically defined by the set of views that it offers.

There are two basic possibilities to define viewpoints. One is filtering where the assumption is that from an existing metamodel the concepts that are relevant for the viewpoint to be defined are selected and after this selections the views are defined. The second possibility is to define an explicit metamodel for the viewpoint and to define all necessary views according to this metamodel. The two options are actually not that different from each other. If a filter is defined for a specific metamodel, one can easily create a separate metamodel by deleting all concepts that are not involved with respect to the defined filter. However, depending on how the filter definition looks like the resulting metamodel might not be very meaningful and model validation for this metamodel might be not very meaningful either. If one is forced into actually defining an explicit metamodel the result might look better because in the definition process of the metamodel missing links or missing information might become obvious.

However, regarding filtering the exchange of information between viewpoints is more straight forward. All that has to be done is to pass on the complete model instances among the viewpoints (for simplicity we assume that only one stakeholder is active on the model instance at one point in time). However, the side effects the different viewpoints might have on each other might be difficult to see. Additionally, information hiding is not really possible and therefore there is no guarantee that a stakeholder is only able to see and manipulate the information that he or she should actually see and manipulate.

4.5 Integration of Methodology Artefacts

We use an example from the Prometheus methodology to illustrate how to integrate model artefacts of a certain agent methodology and tool support into Bochica. The example is based on so called *scenarios* of the Prometheus methodology. A Prometheus scenario is used during the system specification phase to identify relevant entities of the system. A scenario describes a typical workflow of the SUC in terms of *steps*. This also includes the involved roles, goals, perceptions, etc. In later development phases the identified entities are used to derive agent types. Bochica does not natively support the user in identifying those entities. An extension of the Bochica framework for the Prometheus way of identifying those entities is depicted in Figure 5. It consists of (i) the user interface and tool support and (ii) the underlying metamodel which extends the Bochica methodology artefact concept. By extending the Bochica metamodel, the model artefacts become part of the framework. Both, the GUI as well as the metamodel extension, are contributed as an Eclipse-based plug-in to Bochica. Moreover, the Prometheus extension would benefit from the strong conceptual core of the Bochica framework and the mature code generation facilities.



Handle new order scenario

Steps of the sceanrio

Modify the steps after selecting an item from the summary table at the bottom.
To change step order, drag N Drop the item in the summary table.

Type: Goal

Name: HandleNewOrderGoal

Description: User assigns new order.

Roles: Select, or type then clicl Production

Data read: Select, or type then clicl

Data written: Select, or type then clicl Orders

Fig. 5. This figure depicts an example of how BOCHICA might be extended for the Prometheus methodology

5 Case Study

Virtual worlds play an increasingly important role for many application domains. Besides entertainment, they are used for serious applications like digital engineering and for training employees in virtual environments before a product or plant has been actually built. As of today, realistic and flexible behavior of agents in virtual worlds is usually simulated by triggered script sequences which create the illusion of intelligent behavior for the user. In the research project *Intelligent Simulated Realities* (ISReal) our research group developed a simulation platform based on Semantic Web technology for bringing intelligent behavior into virtual worlds [43]. The basic idea of ISReal was to use Semantic Web technology to extend purely geometric objects with ontological information (OWL-based; e.g. concept door links two rooms and can be open or closed) and specify their functionality by semantic service descriptions (OWL-S-based; e.g. open and close door services), called *object services*. Intelligent agents perceive this information, store it in their knowledge base, and use it for reasoning and planning. An object service is grounded in a service implementation which invokes according animations or simulation modules. Before we discuss the BOCHICA extensions for developing intelligent ISReal avatars, we introduce the main components of the ISReal platform.

Global Semantic Environment. The *Global Semantic Environment* (GSE) maintains the global ontological facts of the virtual world. It is responsible for (i) executing object services (e.g. checking the pre-condition and invoking the service grounding), (ii) updating facts (e.g. when a door gets closed), and (iii) handling queries (e.g. SPARQL).

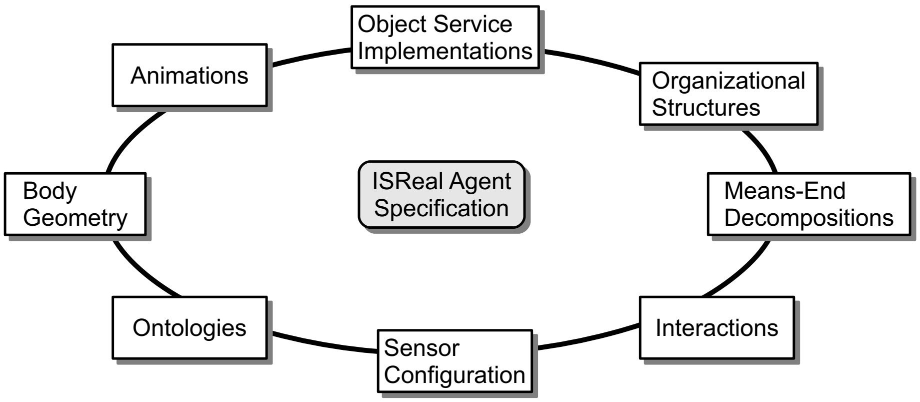


Fig. 6. Overview of the artifacts that encompass an ISReal agent configuration

Agent Environment. The ISReal agent environment defines interfaces for connecting 3rd party agent execution platforms to the ISReal platform (we currently use Jack, Jadex, and the Xaitment[9] game AI engine). Every ISReal agent is equipped with a *Local Semantic Environment* (LSE) which is an agent's local knowledge base. The LSE stores the perceived information and enables the agent to reason about it. Moreover, the LSE is equipped with an AI planner.

Graphics Environment. The user interface of the ISReal platform is realized by an XML3D[10]-enabled standard Web browser. The 3D scene graph is part of the browser's *Document Object Model* (DOM) and can be manipulated using Java Script. It also contains RDFa[11]-based semantic annotations of the 3D-objects such as the concept URI, the object URI, and the semantic object service URIs. Moreover, we extended the browser with an agent sensor which allows agents to perceive the annotated 3D objects.

5.1 Intelligent ISReal Agents

The design of a 3D simulation, like the simulation of workflows of a virtual production line, is a complex endeavor and involves many different stakeholders. In ISReal, agent technology is used to simulate the behavior of human entities in the virtual environment. Figure 6 shows the artifacts that encompass a typical ISReal agent configuration for the development of a typical ISReal scenario. An intelligent ISReal agent consists of (i) the body geometry and animations, (ii) semantic annotations, (iii) a sensor component, (iv) the agent that processes the perceptions and controls the body, and (v) an OWL-based knowledge base. An agent's body geometry is part of the scene graph like any other annotated object. The geometry and animations are developed using state-of-the-art 3D modeling tools.

[9] `http://www.xaitment.com/`
[10] `http://www.xml3d.org`
[11] `http://www.w3.org/TR/xhtml-rdfa-primer/`

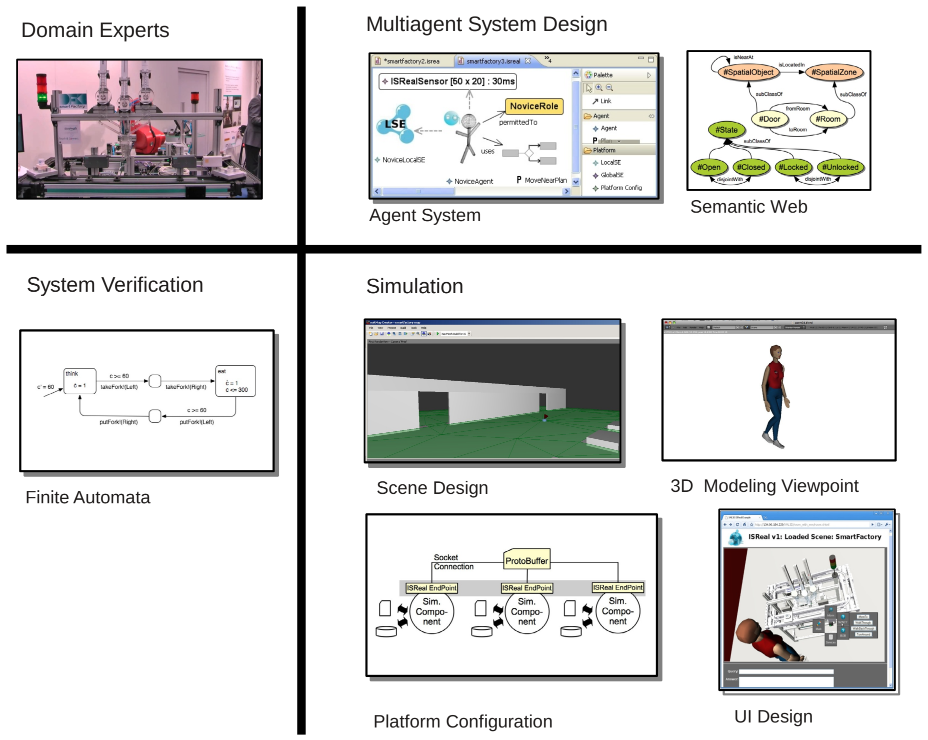


Fig. 7. Viewpoints of a typical ISReal scenario

5.2 ISReal Viewpoints

For the design of a typical ISReal scenario we distinguish 4 different stakeholder for which we foresee viewpoints (see Figure 7). The different viewpoints are summarized as follows:

Domain Experts. The viewpoint of the domain expert depends on the considered system. There might be several domain experts (e.g. mechanical engineer, electrical engineer etc.).

Multiagent System Design. The viewpoint of the designer of the multiagent system and the individual agents is the main viewpoint that is of interest for the research and development of agent technologies. For now we only foresee one viewpoint for this aspect but if the scenarios get more complex and the number of engineers involved becomes larger we plan to specialize this viewpoint according to the ideas presented in Section 4.4. For now we distinguish two major aspects in this viewpoint.

Agent System: This aspect defines the views of the agent design expert to the system for specifying the autonomous entities of the system. This encompasses views for the specification of (i) interactions, (ii) organizational structures, (iii) means-end decompositions, and (iv) agent types. We use the Bochica framework for agent-related aspects.

Semantic Web: The Semantic Web viewpoint defines the views required by Semantic Web experts. This includes a view for specifying concept and object ontologies, as well as a view for defining semantic service descriptions. The ontologies and services have to be able to answer the questions of the end user to the system. Usually, Semantic Web experts use tools like Protegé[12].

Simulation. The simulation viewpoint covers views for other aspects which are relevant for a simulation.

3D Modeling: The views of the 3D modeler viewpoint are tailored to the needs of computer graphics experts. This includes views for geometric design of objects and their animations. Computer graphics experts use tools like Cinema 4D[13] or Blender[14]. Animations have to be modular and fit to the needs of other components (e.g. an agent's capabilities to interact with its environment).

Simulation/Scene Design: The set of simulation artifacts created by the different viewpoints is not sufficient for creating a simulation. The simulation designer viewpoint provides views for integrating the artifacts into one virtual simulation environment. For example, so called *navmeshs* which support the path finding of an agent have to be computed. Moreover, several other aspects have to be modeled to get a running simulation.

Platform Configuration: The platform configuration viewpoint focuses on the configuration of the simulation components that make up the ISReal platform. The configuration of the ISReal platform has to meet the requirements of the modeled simulation.

UI Design: The *User Interface* (UI) design viewpoint defines the views for defining the user interface to the simulation. Since the ISReal UI is based on XML3D, ISReal UI designers use tools for ordinary Web/HTML user interface design.

System Verification. The verification viewpoint provides views for modeling the system as hybrid automata. Hybrid automata are used by engineers to verify properties of the system.

5.3 Example Adaptation

This section provides an example of how the views of the agent viewpoint can be adapted by the Bochica adaptation process to the needs of the application domain of agents in semantically-enhanced virtual worlds. Figure 8 depicts an

[12] `http://protege.stanford.edu`

[13] `http://www.maxon.net/products/cinema-4d-studio/`

[14] `http://www.blender.org`

Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
• Use core DSL for modeling the SUC • Apply base transformation(s) for rapid prototyping • Evaluate target platform(s) • Specify development process • Configure BOCHICA for the development process and target environment	• Refine SUC models • Select base transformation • Apply base transformation • Collect bottom-up requirements • Create extension model • Create extension transformation • Define custom views	• Refine SUC models • Apply base and ext. transformations • Refine code • Collect bottom-up requirements • Refine extension model • Refine extension transformation • Incorporate custom views, tools, and wizards	• Refine SUC models • Apply base and ext. transformations • Refine code • Collect bottom-up requirements • Refine extension model • Refine extension transformation	• Refine...

Fig. 8. Example adaptation process

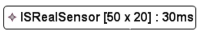
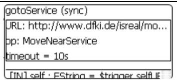
Notation	Concept	Description
	ISRealAgent	• Concept URI • Avatar URI
	LocalSE	• Local ontologies • Local object services • Configuration
	GlobalSE	• Global ontologies • Global object services • Configuration
	ISRealPlatformConf	• Configuration of platform components • Start-up configuration
 ✧ ISRealSensor [50 x 20] : 30ms	ISRealRaySensor	• Resolution • Update rate
 gotoService (sync) URL: http://www.dfki.de/isreal/mo... op: MoveNearService timeout = 10s [IN] self : EString = $trigger.self[	InvokeWS	• Service URI • Parameters • Timeout

Fig. 9. ISReal-extension model for BOCHICA

example for the BOCHICA adaptation process introduced in Section 4.1. The initial iteration is used evaluate and choose the required methods and tools for the problem at hand. For this purpose, a simple prototype of the system is created and mapped to different target platforms. After the tool stack and and the target platform has been defined, the model of the system is gradually refined. Relevant aspects which cannot be modeled using the BOCHICA core DSL are candidates for conceptual extensions of the framework. Moreover, an extension transformation is created which extends an existing base transformation with additional mappings for the new concepts. Thus, the customized framework is able to better close the gap between design and code with each iteration.

Figure 9 depicts an overview of concepts that have been introduced as part of the adaptation process to the Bochica framework. The first column depicts the graphical representation of the concepts. The second column shows the names of the concepts and the third column depicts an overview of the ISReal-specific properties. For example, the `ISRealAgent` concept has an additional ontological concept and a graphical avatar (identified by URIs). The `LocalSE` is the agent's Semantic Web-based knowledge base. It can be configured with OWL-based ontologies, and OWL-S-based object service descriptions. Moreover, there are concepts for configuring an agent's sensor (concept `ISRealRaySensor`) and for orchestrating ISReal object services (concept `InvokeWS`). Two ISReal-specific views have been defined that make use of those concepts:

- **ISReal Agent View:** The ISReal agent diagram extends the Bochica agent diagram with ISReal-specific model elements like the `ISRealRaySensor` and the `LocalSE`. Moreover, it uses the ISReal notation.
- **ISReal Platform View:** The ISReal platform view enables the agent modeler to configure the agent platform ISReal platform. For example, the ontologies of the GSE and the components of the ISReal platform can be configured.

6 Conclusion

In this paper we presented a methodological approach to model-driven design of MAS. We briefly described the Bochica framework for model-driven design of MAS and showed how MAS design methodologies for MAS that are already available from literature can be connected to Bochica. We introduced the RUP methodology for iterative system development and demonstrated how it can be adapted to serve in the context of MAS design. The different stakeholders in MAS design were briefly discussed and the concepts of views and viewpoints were introduced to support the different stakeholders in the best manner possible. Although right now it is possible for system engineers to work on the modeling tool and provide adapted versions that can support the different needs of the different stakeholders, more flexibility in defining views and viewpoints is desirable that users with little or even no system engineering background can at least adapt the existing views with respect to their personal needs. Dynamic creation of viewpoints would by non experts would be desirable, too, but is nothing that could be easily achieved with the technology that is available today.

Acknowledgements. This work has been partially funded by the German Federal Ministry of Education and Research through ICT Projects Collaborate3D (Scalable Intelligent Support and Reliability for Collaboration in the Future 3D Internet, FKZ 01IW11002) and ViBaM (Viewpoint-Based Modeling, FKZ 01QE1106C) which is running in the context of the European Eurostars Program (E!5529). The authors wish to acknowledge EUREKA and the Commission for their support. We also thank the reviewers for their valuable comments.

References

1. Warwas, S., Fischer, K., Klusch, M., Slusallek, P.: Bochica: A model-driven framework for engineering multiagent systems. In: ICAART 2012 - Proceedings of the 4th International Conference on Agents and Artificial Intelligence, Vilamoura, Algarve, Portugal, February 6-8, vol. 1, pp. 109–118 (2012)
2. Warwas, S., Hahn, C.: The DSML4MAS development environment. In: 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, May 10-15, vol. 2, pp. 1379–1380 (2009)
3. Warwas, S., Klusch, M.: Making multiagent system designs reusable: A model-driven approach. In: Proceedings of the 2011 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2011, Campus Scientifique de la Doua, Lyon, France, August 22-27, pp. 101–108 (2011)
4. Royce, W.W.: Managing the development of large software systems: concepts and techniques. In: Proceedings of the 9th International Conference on Software Engineering, ICSE 1987, pp. 328–338. IEEE Computer Society Press, Los Alamitos (1987)
5. Boehm, B.W.: A spiral model of software development and enhancement. Computer 21(5), 61–72 (1988)
6. Kruchten, P.: The Rational Unified Process an Introduction, 3rd edn. Addison-Wesley (November 2005)
7. Zambonelli, F., Jennings, N.R., Wooldridge, M.J.: Developing multiagent systems: The Gaia methodology. ACM Transactions on Software Engineering and Methodology (TOSEM) 12(3), 317–370 (2003)
8. Padgham, L., Winikoff, M.: Developing Intelligent Agent Systems: A Practical Guide to Designing, Building, Implementing and Testing Agent Systems. John Wiley & Sons (2004)
9. Henderson-Sellers, B., Georgini, P. (eds.): Agent-Oriented Methodologies. Idea Group Publishing, Hershey (2005)
10. Weiß, G., Jakob, R.: Agentenorientierte Softwareentwicklung: Methoden und Tools. Springer (2004)
11. Giorgini, P., Mylopoulos, J., Sebastiani, R.: Goal-oriented requirements analysis and reasoning in the tropos methodology. Eng. Appl. Artif. Intell. 18(2), 159–171 (2005)
12. Sterling, L., Taveter, K.: The Art of Agent-Oriented Modeling. The MIT Press (2009)
13. Wooldridge, M.J., Jennings, N.R., Kinny, D.: The Gaia methodology for agent-oriented analysis and design. Autonomous Agents and Multi-Agent Systems 3(3), 285–312 (2000)
14. Gómez-Sanz, J.J.: Modelado de Sistemas Multi-agente. PhD thesis, Universidad Complutense de Madrid. Facultad de Informatica (2002)
15. Pavón, J., Gómez-Sanz, J.: Agent oriented software engineering with INGENIAS. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS (LNAI), vol. 2691, pp. 394–403. Springer, Heidelberg (2003)
16. Pavón, J., Gómez-Sanz, J., Fuentes-Fernández, R.: The INGENIAS Methodology and Tools. In: Agent-Oriented Methodologies, pp. 236–276. IGI Global (2005)
17. Garijo, F.J., Gómez-Sanz, J.J., Massonet, P.: The MESSAGE methodology for agent-oriented analysis and design. In: Henderson-Sellers, B., Giorgini, P. (eds.) Agent-Oriented Methodologies, pp. 203–235. IGI Global (2005)

18. Fuentes-Fernández, R., Gómez-Sanz, J.J., Pavón, J.: Integrating agent-oriented methodologies with uml-at. In: Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), pp. 1303–1310. ACM (2006)
19. Gascueña, J.M., Fernández-Caballero, A.: Prometheus and INGENIAS agent methodologies: A complementary approach. In: Luck, M., Gomez-Sanz, J.J. (eds.) AOSE 2008. LNCS, vol. 5386, pp. 131–144. Springer, Heidelberg (2009)
20. Fernández-Caballero, A., Gascueña, J.M.: Developing multi-agent systems through integrating prometheus, INGENIAS and ICARO-T. In: Filipe, J., Fred, A., Sharp, B. (eds.) ICAART 2009. CCIS, vol. 67, pp. 219–232. Springer, Heidelberg (2010)
21. DeLoach, S.A.: Engineering organization-based multiagent systems. In: Garcia, A., Choren, R., Lucena, C., Giorgini, P., Holvoet, T., Romanovsky, A. (eds.) SELMAS 2005. LNCS, vol. 3914, pp. 109–125. Springer, Heidelberg (2006)
22. Garcia-Ojeda, J.C., DeLoach, S.A., Robby, Oyenan, W.H., Valenzuela, J.: O-maSE: A customizable approach to developing multiagent development processes. In: Luck, M., Padgham, L. (eds.) AOSE 2007. LNCS, vol. 4951, pp. 1–15. Springer, Heidelberg (2008)
23. DeLoach, S.A., Garcia-Ojeda, J.C.: O-MaSE: a customisable approach to designing and building complex, adaptive multi-agent systems. Int. J. Agent-Oriented Softw. Eng. 4(3), 244–280 (2010)
24. Firesmith, D., Henderson-Sellers, B.: The OPEN Process Framework: An Introduction. Addison-Wesley (2001)
25. Padgham, L., Winikoff, M., Poutakidis, D.: Adding debugging support to the Prometheus methodology. Engineering Applications of Artificial Intelligence 18(2), 173–190 (2005)
26. Zhang, Z., Thangarajah, J., Padgham, L.: Automated unit testing for agent systems. In: Proceedings of the 2nd International Working Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2007), pp. 10–18. INSTICC Press (2007)
27. Zhang, Z., Thangarajah, J., Padgham, L.: Automated unit testing intelligent agents in PDT. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), pp. 1673–1674. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS) (2008)
28. Winikoff, M.: Defining syntax and providing tool support for Agent UML using a textual notation. International Journal of Agent-Oriented Software Engineering 1(2), 123–144 (2007)
29. Padgham, L., Thangarajah, J., Winikoff, M.: AUML protocols and code generation in the Prometheus design tool. In: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), pp. 1374–1375. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS) (2007)
30. Cossentino, M.: From requirements to code with the PASSI methodology. In: Henderson-Sellers, B., Giorgini, P. (eds.) Agent-Oriented Methodologies, pp. 79–106. Idea Group Publishing, Hershey (2005)
31. Giunchiglia, F., Mylopoulos, J., Perini, A.: The Tropos software development methodology: Processes, models and diagrams. In: Giunchiglia, F., Odell, J.J., Weiss, G. (eds.) AOSE 2002. LNCS, vol. 2585, pp. 162–173. Springer, Heidelberg (2003)
32. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems 8(3), 203–236 (2004)

33. Penserini, L., Perini, A., Susi, A., Mylopoulos, J.: From stakeholder intentions to software agent implementations. In: Martinez, F.H., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 465–479. Springer, Heidelberg (2006)
34. Morandini, M., Nguyen, D.C., Perini, A., Siena, A., Susi, A.: Tool-supported development with tropos: the conference management system case study. In: Luck, M., Padgham, L. (eds.) AOSE 2007. LNCS, vol. 4951, pp. 182–196. Springer, Heidelberg (2008)
35. Morandini, M.: Goal-Oriented Development of Self-Adaptive Systems. PhD thesis, University of Trento. International Doctorate School in Information and Communication Technologies (2011)
36. Cossentino, M., Gaud, N., Hilaire, V., Galland, S., Koukam, A.: Aspecs: an agent-oriented software process for engineering complex systems. Autonomous Agents and Multi-Agent Systems 20(2), 260–304 (2010)
37. Lind, J.:The MASSIVE Method. LNCS (LNAI), vol. 1994. Springer (2001)
38. Hübner, J.F., Sichman, J.S., Boissier, O.: Moise+: towards a structural, functional, and deontic model for mas organization. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1, AAMAS 2002, pp. 501–502. ACM, New York (2002)
39. Vázquez-Salceda, J., Dignum, V., Dignum, F.: Organizing multiagent systems. Autonomous Agents and Multi-Agent Systems 11(3), 307–360 (2005)
40. Fipa design process documentation and fragmentation working group
41. OMG: Model driven architecture, `http://www.omg.org/mda/`
42. Akehurst, D.H., Kent, S., Patrascoiu, O.: A relational approach to defining and implementing transformations between metamodels. Software and Systems Modeling 2(4), 215–239 (2003)
43. Kapahnke, P., Liedtke, P., Nesbigall, S., Warwas, S., Klusch, M.: ISReal: An open platform for semantic-based 3D simulations in the 3D internet. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part II. LNCS, vol. 6497, pp. 161–176. Springer, Heidelberg (2010)

A Norm-Governed Holonic Multi-agent System Metamodel

Patrizia Ribino[1], Carmelo Lodato[1], Salvatore Lopes[1], Valeria Seidita[2,1], Vincent Hilaire[3], and Massimo Cossentino[1]

[1] Istituto di Reti e Calcolo ad Alte Prestazioni,
Consiglio Nazionale delle Ricerche, Palermo, Italy
{cossentino,c.lodato,s.lopes,ribino}@pa.icar.cnr.it

[2] Dip. di Ingegneria Chimica Gestionale Informatica Meccanica,
University of Palermo, Italy
valeria.seidita@unipa.it

[3] System and Transport Laboratory,
University of Technology of Belfort Montbéliard, France
vincent.hilaire@utbm.fr

Abstract. Modeling and designing systems that require a high level of coordination, control and automation is a very difficult task. The problem is the lack of design processes able to cover all the features these systems present. This paper presents an extension of the ASPECS metamodel for supporting organizational and normative principles and it allows to define models not only from an holonic agent viewpoint but also from a normative organization perspective. Moreover, our work emphasizes and makes it explicit the norms that regulate the structural, behavioral and finally adaptive aspect of an organizational system. The extended metamodel was experimented creating a Virtual Enterprise model for the optimization of distributions inside the logistic districts. This organizational model is implemented using JaCaMo.

1 Introduction

Nowadays, a lot of researchers in the field of artificial intelligence and intelligent systems aim to develop software systems able to act in full autonomy such as a human beings do in reaching their objectives. During their daily activities, human beings pursue multiple goals that sometimes interleave and overlap; in doing that, they often communicate and coordinate with other entities of the world they live. We are far from having tools to create systems completely acting as they were in a daily "human routine". Nevertheless, literature proposes a way for developing goal-driven systems using the knowledge these systems have of their environment in order to react to environment changes. The ability of coordinating, controlling and making autonomous the activities of all the different involved entities is a strong requirement for this kind of systems. These issues can be faced with the use of the Multi Agent System (MAS) design paradigm and with organizational models. The latter is covered by only a restricted set of agent oriented design processes[1] of which only few cover the whole design process life cycle, from analysis to implementation. To the best of our knowledge, among the agent-oriented processes only ASPECS [6] manages abstractions, such as holon, group and goal, for modeling and implementing organizational structures like holarchy. Instead, among organizational models, only $\mathcal{M}$oise+ [13] and OMNI [26] address the normative aspect of a multi-agent organization.

The novelty introduced by our work is merging the strength of ASPECS and $\mathcal{M}$oise+ in order to create a complete support for developing MASs structured organizations, such as holarchies, ruled by norms. Actually ASPECS does not include the possibility of designing norm-based systems. The need of introducing norms arose from a design requirement, we needed the possibility of modeling constraints in form of institutional rules (Norms) defined outside the agents. Defining external rules (like social rules, laws, company procedures, etc. . .) allows to face all the problems related to management, coordination and control of different holons. The result was an extension of the ASPECS metamodel in order to include all the elements providing abstractions for managing the normative issues along with the definition of some new norms that regulate the structural, behavioral and adaptive aspect of an organization. Moreover, we instantiated this new metamodel in a specific logistics business model in order to create an optimized representation of the distribution processes inside a supply chain. We have implemented this model as a norm-governed holarchy using Jason [5], $\mathcal{M}$oise+ and Cartago[22], unified in the JaCaMo framework [24].

The rest of the paper is organized as follows. In Section 2 an overview on the theoretical background of the work is presented. Section 3 is the core of the paper. It presents the extended metamodels along with the definition of norms we have introduced and the conceptual mapping among the elements of the Agency Domain and JaCaMo metamodel. In Section 4 we show the addressed case study. Finally some conclusions are presented in Section 5.

2 Theoretical Background

An agent is an *autonomous*, *reactive* and *proactive* entity that pursues individual goals interacting with the environment and others agents by means of *social ability* [28]. A multi-agent system [9] (MAS) is a software providing a tool to model and reproduce the interaction and social structures observed in real world organizations. It allows to adapt human organizational patterns in agent-based systems that become a virtual counterpart of real organizations.

As well-known, there are different kinds of organizational schema, such as hierarchies, holarchies, teams, coalitions and so on. Each organizational schema is usually defined by means of roles adopted by an agent, relationships, rules and norms defining the agents behavior and organizational structure.

In this paper we adopt holarchies as an organizational structure of the agents societies. The concept of Holarchy adopted as an Enterprise model has its origin

[1] We use the term design process and methodology as synonyms because here it is not important to highlight the differences among them.

from the work of Koestler [16]. During his research on self-organization in biological systems, Koestler discovered nested hierarchies of self-replicating structures (*holarchies*). He coined the term *holon* to describe the elements of such systems.

A holon is, commonly, defined as a self-similar structure composed of holons as sub-structures. For this reason, it can be seen from different perspectives, either as an autonomous *atomic* entity or as an *organization* of holons. A holon is a whole-part composed of other holons and at the same time, a component of a higher level holon. A holon acts basically as an autonomous entity, although cooperating to form self-organizing hierarchies of subsystems (such as the cell/tissue/organ/system hierarchy in biology) in order to achieve the goals of the holarchy. In addition, holons can simultaneously belong to different super-holons and can be regulated by rules. These rules not only allow to define a system as a holon with an individuality of its own but also to determine its structural configuration, functional patterns and behavioral regulations [25].

Holonic systems, while modeling complex systems, are able to efficiently manage their resources and to adapt themselves to changes occurring in the environment. A useful way to implement holarchies in software system is by means of the Holonic Multi-Agent System (HMAS) paradigm. As shown in [10], HMAS paradigm allows to represent a holonic system where individual agents are driven by coordination mechanism according to the cooperation rules of the holon the agent is member of. In HMAS a holon is a set of individual agents organized according to different organizational models (see [10] for more details).

In this paper we use the HMAS and the Virtual Enterprise paradigm to model a holonic framework applied to the cited logistic problem. According to Uliero *et al.* [25] we refer to Virtual Enterprise (VE) as a new organizational form that can be characterized by a collection of geographically apart individuals, groups or entire organizations depending on electronic communications in order to collaboratively work and to provide a service or to realize a common goal.

Multi agent systems can be developed using several frameworks (JADE [2], JADEX [20][27], PRACTIONIST [18] etc.) based on different approaches. In this paper we adopt the JaCaMo approach in order to implement a Holonic Multi-Agent System. JaCaMo [24] is a programming platform that integrates three levels of multi-agent abstractions: an agent programming language (Jason), an organizational model (Moise), and an environment infrastructure (CArtAgO).

Jason [5] is a Java-based interpreter for an extended version of the AgentSpeak language [21], an abstract agent language founded on the BDI(Belief-Desire-Intentions) model. A Jason agent is described by means of a set of plans the agent is able to follow in some situations.

$\mathcal{M}$oise+ [13][14] is an organizational model for MAS which specifies the structural, functional and normative aspect of MAS organizations. Each aspect is defined in a specification set.

CArtAgO [22] is a general purpose framework/infrastructure that allows to program and execute virtual environments for multi-agent systems. It is based on the concept of Artifacts intended as resources and tools dynamically constructed, used, manipulated by agents to support/realise their individual and collective activities.

3 A Norm-Governed Holonic MAS Metamodel

In order to have means for developing norm-governed multi-agent systems structured by holonic organizations we need a metamodel containing all the abstractions to be treated during the phases/activities of the design process devoted to develop such systems. In this section we illustrate the metamodel we created by adding to the ASPECS metamodel all those concepts from $\mathcal{M}$oise+ metamodel useful for modeling MASs under a normative point of view.

The ASPECS metamodel [6] is divided in three parts: the problem, agency and solution domain. The first contains the elements useful for the description of the problem under an organizational point of view. The second domain provides an agent oriented solution to the problem. Finally, the last provides the concepts for the implementation in a specific platform. As also stated in [15], ASPECS is one of the most complex and complete organizational approaches. It covers all the organizational aspects considered in other design processes *"(roles, tasks, plans, goals, organizations, resources, agents and, in this case, holonic structures), rich interactions (communication, protocols, messages) and a formal definition of the domain knowledge (ontology)"*.

Nevertheless, ASPECS does not cover some aspects such as those related to the tasks to be accomplished by the organization and the rules to observe in order to ensure the profitable achievement of the goals of the organization. For these reasons the $\mathcal{M}$oise+ metamodel (deduced from [12]) was taken into account. In particular, we have paid attention to the following $\mathcal{M}$oise+ concepts: the Role constraining the agent's behavior; the Organizational Link regulating the social behavioral part of agents and the group, to which agents belong; the Norms, which rule the set of roles and missions that agents can do.

Our work consists in the definition of a new metamodel that emphases the normative aspect of a HMAS. To do this: (i) we have extended the ASPECS's Problem and Agency Domain metamodels with the previous $\mathcal{M}$oise+ concepts; (ii) we have specialized the concept of Norms into three categories: Behavioral, Structural and Adaptive Norms; (iii) we have mapped these new extended metamodels in the Solution Domain provided by JaCaMo platform. In the following subsections, we explain these three steps. For the sake of clarity, in Fig. 1 and Fig.2, we have differently colored the new concepts to highlight the differences with ASPECS metamodels. In the following, we give a detailed description of new concepts referring to ASPECS metamodel for those not mentioned in this paper.

3.1 Problem Domain Metamodel

The extended Problem Domain metamodel is shown in Fig. 1. According to ASPECS, an *Organization* can be an aggregate of other sub-organizations. Each organization is composed of *Roles* which specify the *Capacities* that should be owned by an agent to play them. *Interactions* between *Roles* define *Scenarios* where each *Role* contributes to the achievement of organizational objectives (*Requirements*). Unlike ASPECS, we highlight that an *Organization* is plunged

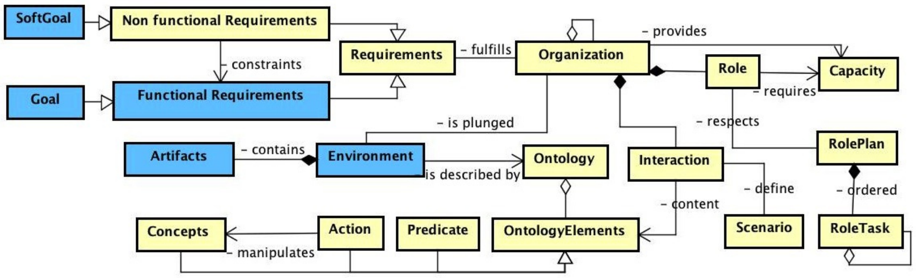


Fig. 1. Problem Domain metamodel

in an *Environment* composed of *Artifacts* that can be either passive elements (e.g. resources) used by agents and normative elements (e.g. social laws) imposed on organizations and their members in order to fulfill their goals. Each element of the *Environment* is described by means of an *Ontology* providing a common vocabulary and a machine-readable knowledge.

In the following we give a brief description of the new concepts we have introduced in the Problem Domain.

Functional Requirements describe the functions the software has to execute. In some context, often also in agent-oriented systems, they are also known as *Capabilities*[1].

Nonfunctional Requirements [1] are seen as constraints or quality requirements of the solution to be adopted.

Goals and **Softgoals** are a specialization of functional and nonfunctional requirements respectively. A *Goal*, representing an actor' strategic interest, can satisfy a system requirement. While *Softgoals* are generally considered as goals for which it is difficult to decide whether they are satisfied or not. In our model we use *Softgoals* to constrain *Goals.*

The **Environment** is a first-class abstraction that provides the surrounding conditions for agents to exist and that mediates both the interactions among agents and the access to resources. The passive components of the system, such as resources and objects, that are shared and used, cooperatively or competitively, by agents to support their activities or norms, rules, physical and social laws that act on the environment or govern its living entities are represented by means of **Artifacts** [19]. For this reason, we see the environment as a set of artifacts that form a context in which agents perform their tasks and pursue their goals. A special kind of artifacts that we considered in this paper are the norms, which will be deeply explained in the next section.

3.2 Agency Domain Metamodel

Several complex dynamic systems, naturally occurring as well as those created by the society, show common features. Such as (i) the composition of entities operating in parallel *(nerve cells into brain, individuals or enterprise in a market*

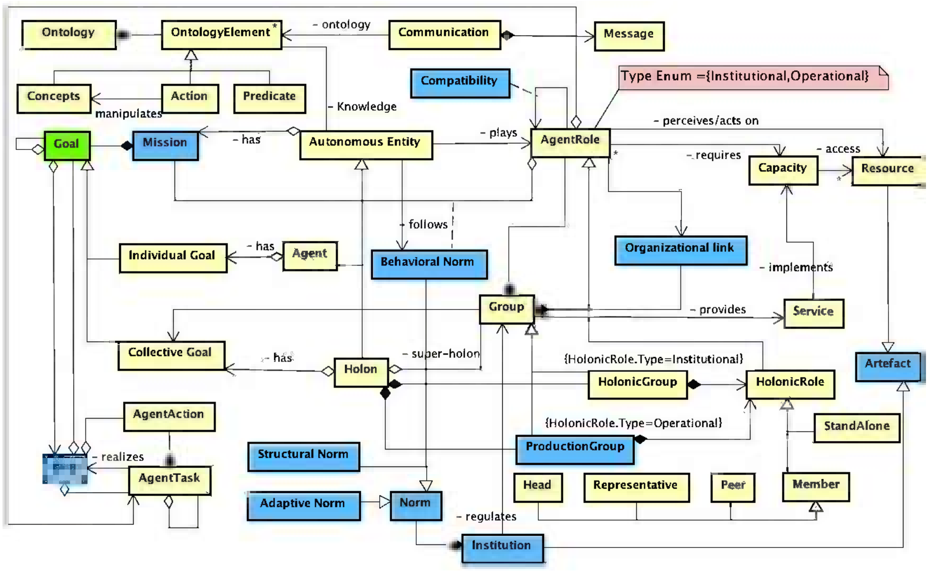


Fig. 2. Agency Domain metamodel

economy, etc...), (ii) different levels of organization (*proteins and lipids form a cell, cells form tissues, tissues form organisms and so on*), (iii) the continuous adaptation of their components through the process of evolution (*adaptation involves the recombination of the component elements or the generation of a new one*), only to name but a few. Adopting an organizational approach to model these kinds of system is a very useful way to represent their structure as well as all the elements indispensable to define a solution.

The Agency Domain metamodel shown in Fig. 2 describes an organizational solution from the agent-oriented perspective along with normative concepts. As ASPECS does, we consider *Holons* the base elements of the organizational solution. The *Holons* are recursively composed of other Holons and, at the same time, each of them is composed of groups. In our extended model we considered two different kinds of *Groups*: the **Holonic Group** and the **Production Group**. At the first level of abstraction, members of the *Holonic Group* play *Institutional Roles* to which are assigned the task to regulate the organizational aspect of the system and to enforce the norms. In that, we accepted the position of V. Dignum *et al.* who say [8] that *Institutional Roles* are roles needed in order to keep the society going. Differently, members of the *Production Group* play *Operational Roles* to which is assigned the task to perform activities necessary to pursue the organizational objectives in accordance with the *Behavioral Norms* and their *Mission.* A **Mission** is "*a set of constraints that the agent must take into account when it wants to execute parts of this task. It defines an allowed behavior as a consistent set of authorization related to goals to be achieved, plans to follow, actions to execute and resource to use*" [12]. A set of missions to which an agent must obey is assigned to each *AgentRole*. A **Plan** is defined as an oriented graph where each node can be a simple *Agent Action* or *Agent Task* or a set of sub-goals. It represents the way to reach the organizational objective. In this context a *Goal* is seen as an aggregate of *Plans*. Roles inside different *Holons* are linked by means of *Organizational Links*. The **Organizational Links** define the way in which the social exchanges between *Agent Roles* occur [12].

The most significant difference compared to ASPECS is the introduction in the Agency Domain of the **Norms**. A general definition of Norm is *an authoritative standard or model*. We have specialized this concept making explicit different kinds of Norm: *Behavioral*, *Structural* and *Adaptive Norms*.

We called **Behavioral Norm** what Boella et.al [3] define "*a principle of right action binding upon the members of a group and serving to guide, control, or regulate proper and acceptable behavior*", similar to the concepts of *Regulative norms* described as *the expected contributions to the social system* [23]. In our model, a *Behavioral Norm* regulates the way a *Agent Role* performs a *Mission*. Two main types of *Behavioral Norms* are *Obligation* and *Permission*. When an **Obligation** is established between an *Agent Role* and a *Mission*, the Autonomous Entity playing the *Agent Role* is obliged to execute the *Mission*. Instead, when the Behavioral Norm is a **Permission** then the Autonomous Entity playing the *Agent Role* can decide to execute the *Mission* or not [12].

The *Structural* and *Adaptive Norms* are instead two new kinds of norm we propose in order to regulate the static and dynamic aspects of an organization separately. The **Structural Norms** define the static structural aspect of the system at the design time, that is the initial composition defined by the designer for the organization to fulfill its objectives. The **Adaptive Norms** govern the state transition of the organization from a given configuration to a new one according to needs emerging from environmental changes. By means of adaptive norms the agent society spontaneously evolves toward another optimal configuration for the new state of the world.

The last element introduced in the model is the *Institution*. **Institutions** [7] provide the social and institutional backbone of the agent society and they are the place where social norms are explicitly specified.

In the following subsection we highlight some general structural norms that a holonic organization must comply with. As regard the adaptive norms, in this paper we provide only a preliminary introduction without discussing any theoretical details that will be argued in another specific work.

Structural Norm

When we want to adopt a solution based on organizations (without going into the details of a methodological approach for holonic organization design), the organizational structure is the first element to be defined. The choice of the appropriate organizational schema is related, first of all, to the global objective of the system. Its performance depends on the way tasks are distributed among individuals, how their responsibilities (assigned to Roles) are defined and how they could be aggregated in groups. For instance, organizational groups can be created as functional units responsible to execute either a process or some of its phases, depending on the interdependence of groups involved in the work flow execution. In the following, we exemplify an organizational schema by means of structural norms. In particular, we distinguish the norms needed for the design of holons from those used for the definition of organizational schema.

The following list shows a sub-set of structural norms that allow us to define a holarchy [17]:

1. A generic holonic structure must contain at least three levels of representation. The level (n) represents a holon as a *whole* with its unique characteristics, the level (n-1) contains the holons subordinated to the previous one, finally, the level (n+1) holon is a *super-holon* containing the level (n) holon (and others if required).
2. A *top holon* is not included in any holon of level(n+1).
3. A *bottom holon* does not include holons of level (n-1).
4. A *stand-alone* holon is a non-member holon. It can be seen as a top and bottom holon at the same time.
5. Holons of the same level cannot be included in each other.
6. The number of holons at level (n) cannot be greater than that of the holons at level (n-1).
7. Holons at level (n) can be part simultaneously of holons at level(n+1).
8. Holons at level (n) that are not decomposable can be brought to a lower level(n-1) by means of virtual holons (see Fig. 5).

While, in order to define the organizational schema such as for example a *moderated group* [11] three roles are necessary. The *Head* role identifies the decision maker of the holon. The *Representative* role is the interface of the holon outside the world. Finally, the *Peer* role identifies the default members, generally they perform tasks assigned by the Head. This organizational schema will be well-formed applying the following structural norms:

1. A moderated group must contain at least one individual playing the *Representative* role.
2. It must contain at least one individual playing the *Head* role.
3. It can include from zero to a generic number of *Peer* players.
4. Head and Peer are exclusive roles.
5. Members belonging to only one super-holon adopt the *Part* status.
6. Members of the moderated group can belong to more than one super-holon, adopting the *Multi-Part* status.
7. The Part status is adopted by default.
8. Part and Multi-Part are exclusive status.

We have adopted these norms to define the structural specification for our case study.

3.3 Conceptual Mapping in a Solution Domain

Designing systems normally results in realizing a possible implementative solution in a given platform. The aim of this section is to give a possible implementative solution to a holonic organization using the platform called *JaCaMo*[4].

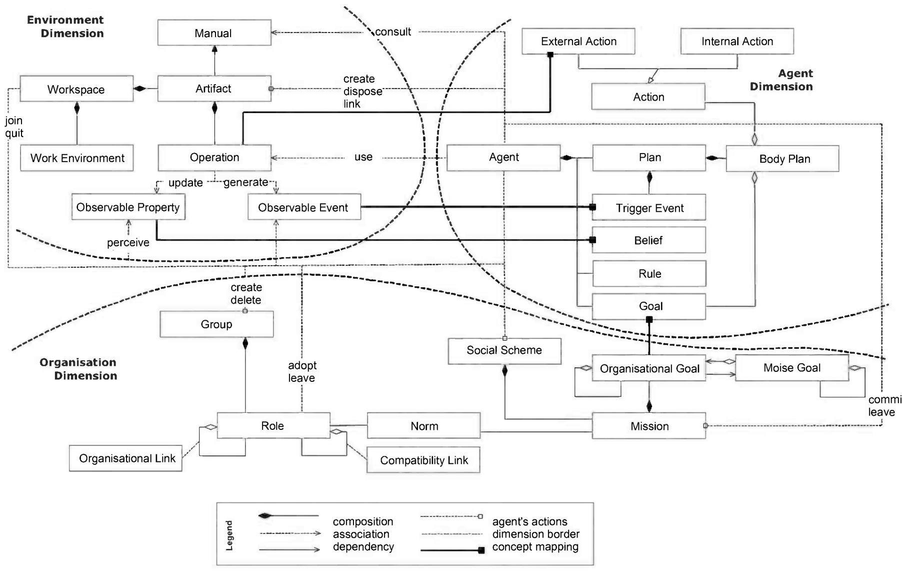


Fig. 3. The JaCaMo Meta-Model adapted from [4]

Among several existing platforms, we have chosen JaCaMo because it natively supports key concepts such as organizations and norms.

JaCaMo is a framework providing a set of programming abstractions that allow us to implement a holonic organization of BDI agents in a shared distributed artifact-based environment. It gives an integrated vision of three fundamental aspects for the implementation of a multi-agent system, such as: the agents belonging to the MAS; the organizational structure on which the MAS is based; the environment in which agents are plunged. Fig. 3 shows an adapted version of JaCaMo metamodel [4].

In *JaCaMo*, an *Agent* is an autonomous entity owning *Beliefs*, *Plans* and *Rules* that allows him to pursue its *Goals*. The *Beliefs* represent the knowledge owned by an agent about itself and the environment in which it lives. The *Rules* are ways to infer new knowledge starting from some *Beliefs*. The *Goals* are the states of the world the agent wants to reach. Finally, the *Plans* are ways to reach goals. The *Trigger Event* defines the circumstances in which a plan should be considered. The *PlanBody* is the core of a plan. It contains *Actions* and others sub-goals to be performed/achieved in order to fulfill the goal a plan was defined for. *Actions* are simple tasks that an agent can perform. There are two kinds of action: *Internal Actions* (that does not produce changes in the environment) and *External Actions* (changing the environment).

An *Agent* interacts with the *Artifacts* (non living entities) in the environment performing *Operations*. An operation generates *Observable Events* and it updates an *Observable Property* of the Artifact.

Agency Domain Element	JaCaMo Element		Code Portion
Holon	Group		<organisational-specification id=[Holon ID]> … </organisational-specification>
Holonic Group	Group		<sub-groups> <group-specification id=[Holonic Group ID] > … </ group -specification> </sub-groups>
Production Group	Group		<sub-groups> < group -specification id=[Production Group ID]> … </ group -specification> </sub-groups>
AgentRole	Role		<role id=[Role Name]></role>
Organiz. Link	Organizational Link		<link from=[Role Name] to=[Role Name] type=[Autority | Acquaitance | Communication] scope= [intra-group | inter-group] extends-sub-groups=[True | False] bi-dir=[True | False]/>
Compatibility	Compatibility Link		<compatibility from=[Role Name] to=[Role Name] scope=[intra-group | inter-group] extends-sub-groups=[True | False] bi-dir=[True | False]/>
Resource	Artifact		class [ArtifactName] extends Artifact { void init() { defineObsProp ([ObservablePropertyName], 0);} … }
Service	Operation		@OPERATION void changeObservableProperty (int PropertyValue) { int c =getObsProperty([ObservablePropertyName]).intValue(); if (PropertyValue > c) updateObsProperty([ObservablePropertyName],PropertyValue);}
Ontology Element	Belief		functor(term1, ..., termn)[annotation1, ..., annotationm]
Comunication	Internal Action		.send([AgentName], [Performative], [Content])
Individual Goal	Goal		! functor($term_1$, ..., $term_n$)
Collective Goal	Organisational Goal		<goal id=[Goal ID]>
Agent	Agent		[AgentName] agentArchClass jmoise.OrgAgent;
Plan	Plan		Triggering Event : Context <- PlanBody.
	Organizational Plan		<plan operator=[sequential | parallel |choice] >… </plan>
Agent Task	Body Plan		PlanBody
Agent Action	Action	Internal Action	. actionName($term_1$, ..., $term_n$)
		External Action	actionName($term_1$, ..., $term_n$)
Mission	Mission		<mission id=[Mission ID] >…</mission>
Behavioral Norm	Norm		<norm id=[Norm ID] type=[Obligation | Permission] role=[Role Name] mission=[Mission ID] />
	Rule		functor($term_1$, ..., $term_n$) :- Logical Expression.
Structural Norm	Formation Constraints on Group, Role, Mission		<role id=[RoleName] min=[0…N] max=[0…N]> <mission id=[Mission ID] min=[0…N] max=[0…N]> …
Adaptive Norm	Belief		adaptiveNorm ([RoleName], [Entry_Condition], [Plan])

Fig. 4. Conceptual mapping among Agency Domain and JaCaMo elements

Finally from the organizational viewpoint, an agent can adopt *Roles* defined into a *Group*. Agents playing different roles can interact each other only if their roles are connected by *Organizational Links*. An Agent can also play two or more compatible roles at the same time. When an agent adopts a *Role* it is committed to a *Mission* by means of *Norms*. A mission is responsible of a set of *Organizational Goals* reachable by means of *Organizational Plans*. The *Social Scheme* groups *Missions* and it defines the functional aspect of an organization.

The table in Fig. 4 shows the conceptual mapping among the Agency Domain and the JaCaMo metamodel elements along with a codified solution. In particular, we want to underline that *Plan* and *Behavioral Norm* elements do not have a unique mapping with the elements of the solution domain. This is due to the dual nature of a holonic MAS we want to implement. In the solution domain, in fact, Plans and Behavioral Norms are defined differently when they refer to the holon as a whole or as a part.

As we previously said, we adopted the HMAS paradigm in order to implement holarchies in software systems. In HMAS a holon is a set of individual agents organized according to an organizational model. When we want to model an HMAS we usually define the *Collective Goals* of the entire holon as well as the *Individual Goals* of the members of the holon (single agents). In our Agency Domain both *Individual* and *Collective Goals* can be reached by means of *Plans* (hereafter *Agency Domain Plan*). In the Solution Domain, the *Agency Domain Plan* concept is associated to two different elements (see Fig. 4): *Plan* and *Organizational Plan.* An *Agency Domain Plan* can be mapped in a *Plan* of the JaCaMo agent dimension in order to define as an agent could reach its own *Goal* (see Fig. 3). The *Agency Domain Plan* can be mapped in an *Organizational Plan* in order to define as an entire holon could achieve its own *Organizational Goal* (see Fig. 3).

As concerns a *Behavioral Norm* (see Fig. 4), it can be translated in the Solution Domain in two different elements: *Norm* and *Rule.* The former regulates the agent's behavior playing *Roles* inside a holonic system. The latter may regulate an agent 's behavior in the environment independent from the Role it plays.

As regards the *Structural* and *Adaptive Norms*, the JaCaMo meta-model does not support natively these kinds of norms. Thus, we have mapped the *Structural Norm* in the formation constraints imposed on Group, Role and Mission elements of the JaCaMo metamodel. We are currently working for the definition of a new element in the solution domain that may directly implement a Structural Norm.

Conversely, we have already defined a way to represent an Adaptive Norm in the Solution domain (see table in Fig. 4). They are codified as a Beliefs with the following specific notation:

$$\textit{adaptiveNorm}\ ([\textbf{RoleName}], [\textbf{EntryCondition}], [\textbf{Plan}])$$

where *[RoleName]* identifies a list of roles to which the adaptive norm can be applied. *EntryCondition* represents a set of environment changes to whom the agent (playing the RoleName) tries to adapt itself. *Plan* define how the agent could adapt itself.

For space concerns, we omit to detail the remaining elements of Fig. 4, which are however easily understood because they have a direct codification in JacaMo framework. In the following section we present our case study.

4 Case Study: Virtual Enterprise for Logistics

The work presented in this paper was carried out under the IMPULSO[2] project and it represents the solution we have studied for it. IMPULSO - *Integrated*

[2] Further information available at `http://www.vitrociset.it` - Section Ricerca& Sviluppo

Multimodal Platform for Urban and extra urban Logistic System Optimization - offers an integrated system for goods management within the logistic districts, for their storage in special metropolitan distribution centers and finally for distribution within the cities. The development of the IMPULSO system was the test-bed for evaluating and assessing the newly created metamodel with all its concepts. Indeed through the enactment of the design activities devoted to instantiate each concept we were able to create the model of the system (the Figures from now on are parts of the artefacts composing such model) on the base of the right specification provided by the metamodel. We experienced the completeness of the proposed metamodel, indeed both the domains contain all the useful concepts for representing the problem we were dealing with and for describing the solution in terms of holons. Moreover we were able to analyze and establish the behavior of each part of the system through the use of the identified norms.

In the following subsections only three artifacts of the development process are illustrated. They deal with the concepts of holon, group, role and norm.

The Holonic Architecture. The whole Impulso System was modeled as a Virtual Enterprise that is a holarchy of collaborative systems, where each system is a holon itself. Each of them is composed of other systems that act according to the same organizational schema, at the same time they perform different functions at lower levels of resolution. For space concerns, we show only a member of Impulso Holarchy: the Yard Management System(YMS).

The YMS deals with goods traffic inside logistic districts. It manages the automatic container loading and unloading by means of the use of AGVs (Autonomous Guided Vehicles) which move independently but are coordinated in accordance to predetermined patterns by a remote control center. Fig. 5 shows the holonic architecture we have designed for YMS. As we can see, the YMS is composed by three holons: the YMC (*Yard Management Central*), the Freight Forwarders and the YMP (*Yard Management Peripheral*). These holons interact to fulfill the goal of their organization, the YMS, although they themselves are autonomous entities with personal objectives. The holonic enterprise framework, which connects enterprise entities, allows information exchange through communication channels and resources management.

Groups, Roles and Norms. In this subsection we define the entire composition of the holonic organization of the YMS (see Fig. 5). In particular, we define its structural and functional aspects correlated to its normative features.

According to the metamodel shown in Fig. 2, there are two aspects that overlap in a holon. The first is the *holonic aspect* that is directly related to the holonic character of the entity, i.e. a holon (super-holon) is composed of other holon members. As Fig. 5 shows, the YMS super-holon is an entity on its own although composed by members. So, the holonic aspect refers how members organize and manage their representative super-holon (i.e. how they form the Holonic Group). We adopted the *moderated group* configuration as organizational structure for the Holonic Group of each super-holon. Thus, each Holonic Group is

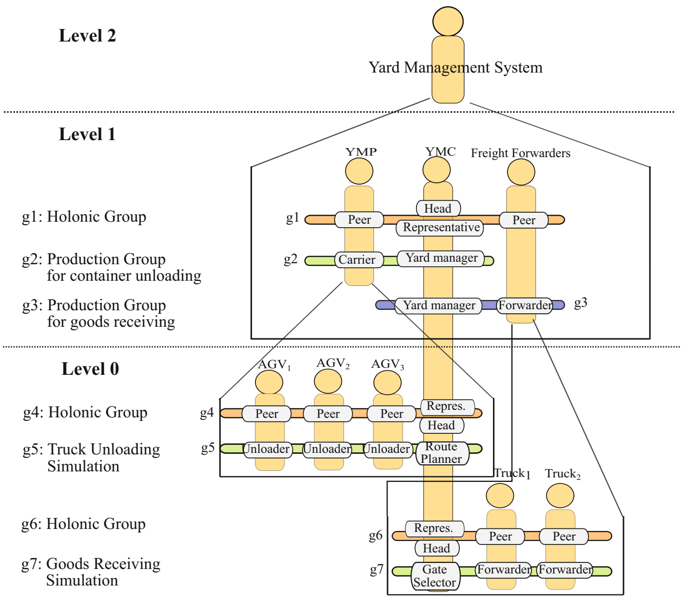


Fig. 5. Roles/Groups of the Yard Management System

created according to the structural norms defined in the section 3.2 which related to the moderate group formation. The second aspect of the holon is related to the problem the members are trying to solve (we will call this the production aspect in order to maintain a uniform nomenclature). The production aspect refers to how members of the holon are organized to pursue their goals according to the global objective of their super-holon. This holonic representation, by means of holonic and production groups, allows to clearly distinguish the different features and functionalities to be attributed to each member.

In the following, we describe only the lowest level of abstraction of the YMS architecture. At this final (finer grained) level of decomposition (see Fig. 5), the holons are represented by groups of agents which play institutional and operational roles at the same time. We focus only on operational roles and production groups, since the institutional roles of the holonic groups have been already described in the section 3.2.

For the simulated scenario, we have defined two production groups (*Truck Unloading Simulation* and *Goods Receiving Simulation*) of two high-level holons. The *Truck Unloading Simulation* is a group formed by the Unloader and Route Planner roles. The Route Planner can be played by YMC agents, which have the capacity to perform the related task. While the Unloader is played by AGV agents which emulate the behaviour of real automated guided vehicles. The *Goods Receiving Simulation* group is formed by the Forwarder and Gate Selector roles. The Forwarder is a role adopted by agents emulating the behaviour of the trucks. As we can see from Fig. 5, the Route Planner player in the Truck Unloading Simulation group adopts the role of Gate Selector in the Goods

Receiving Simulation group, at the same time. This is allowed by the structural norm concerning the multi-part status previously defined (see section 3.2).

In the following list we only show some structural norms we have defined for regulating the formation of Truck Unloading Simulation groups. Then, we present their codification in JaCaMo framework. We avoid to list them all because the presented subset provides enough information in order to understand the purpose of the structural norms.

1. The Route Planner role must be played by at least one agent.
2. The Route Planner role can be played by only one agent.
3. The Unloader role must be played by at least one agent.
4. The Unloader role can be played by at most 10[3] agents.
5. At least one Truck Unloading Simulation group must be active.
6. It can not be created more than 10 Truck Unloading Simulation groups simultaneously.

The norms related to the Route Planner (i.e: norms 1. and 2.) and the Unloader (i.e: norms 3. and 4.) role are codified respectively in

```
<role id="route_Planner" min="1" max="1"/>
<role id="unloader" min="1" max="10"/>.
```

While the last two rules concerning the Truck Unloading Simulation groups (i.e: norms 5. and 6.) are translated into

```
<group-specification id="truck_Unloading_Simulation" min="1" max="10">.
```

As concern the functional aspect of Yard Management System, it is defined by set of plans and missions the agents can commit into a Social Scheme (see Fig.3). It describes how an organization can achieve its global goals. The Fig.6 gives an overview of the Social Scheme of the organization shown as a goal decomposition tree. The root goal of the Yard Management System is sorting out goods toward metropolitan centers. To do that, the members of two production groups can play the permitted roles according to the structural norms and commit to some missions according to behavioral norms described below. Groups perform their activities independently. Holonic groups are responsible for managing their respective production groups and their coordination. For everything else, the Fig.6 is self explanatory.

In the following we show those behavioral norms, which represent JaCaMo Norm elements, according to following template:

```
norm<id> : type=[Obligation | Permission]
role=<RoleName> mission=<MissionName>
```

As we have previously said, these norms impose agents to commit to certain missions when they choose to play a role. Some of them are reported below:

[3] This is a constraint of the project because the AGVs are costly resources.

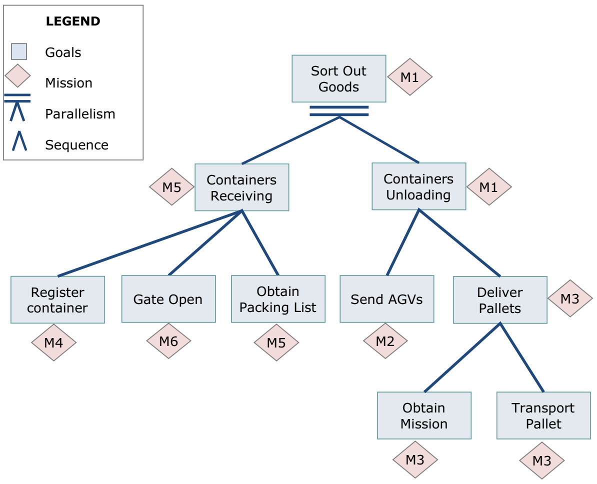


Fig. 6. The functional view of Truck Unloading simulation group represented by means of a goal/mission decomposition tree

```
norm 1: type=Obligation role=unloader
        mission=AgvMission
norm 2: type=Permission role=forwarder
        mission=ForwarderMission
norm 3: type=Permission role=representative
        mission=ManagementMission
norm 4:  type=Permission role=representative
        mission=RecruitmentMission
```

Finally, we have introduced some adaptive norms that allow us to regulate the dynamic evolution of the Yard Management System. We have defined a set of norms that allow the adaptation of the holon to perceived environmental changes. Three examples are:

– If the workload grows beyond some limit (for instance a new truck arrives to be unloaded), the Representative holon creates a new Truck Unloading Simulation group.
– If the workload decreases (for instance unloading operations of a truck are over), Truck Unloading Simulation groups are removed proportionally by the Representative.
– If all role-players leave Representative roles an election has to be made for new players.

According to the definition given in section 3.3, the codified solution of the listed adaptive norms are:

1. *adaptiveNorm*(representative, workload(W) & W >Treshold, @HolonReorg +! createUnloadingSimGroup <– .createGroup(GroupSpec, IDHolon, ID))

2. *adaptiveNorm*(representative, workload(W) & W <Treshold2, @HolonReorg2 +! removeUnloadingSimGroup <– .removeGroup(IDHolon, ID))

3. *adaptiveNorm*(Role, violated(RepresentativeStructNorm), @HolonReorg3 +! newElection <– vote(Player)), where RepresentativeStructNorm is

   ```
   <role id="representative" min="1" max=N />
   ```

   that is the codification of the structural norm 1. of section 3.2 related to moderated group formation.

We want to point out that all upper-case terms are variables that can assume different values during the running of the system. This means that the same adaptive norm can be triggered by different conditions. For example for the first norm, the threshold is a variable value according to the number of created group. In fact the first time the threshold has a defined value according to the amount of work the members of Truck Unloading Simulation group are able to perform. Thus when this norm is applied not only a new group is created but a new value of threshold is automatically defined. This avoids the creation of groups not necessary. Analogous considerations can be made about the second norm. Moreover, if the second norm is applied, the number of Truck Unloading Simulation groups can not become less than one, because it violates the above defined structural norm (i.e: norm 5.) of this production group.

Instead, as concerns the last norm, it is different from the previous ones because it can be adopted by different roles (Role in this norm is a variable) and it is triggered by a violation of a structural norm related to the formation of a moderated group. Thus when this violation occurs, all agents, playing roles inside holons, vote a player from a list of possible candidates according with some defined criteria.

5 Conclusions

In order to solve problems and engineering systems related to fields in which a high level of coordination, control and automation is needed we propose an extension of the ASPECS metamodel obtained by introducing some new concepts such as Norms. Norms are used to regulate holons' behavior, these norms separately deal with the behavioral aspect of the holonic members from the organizational one. From the agents viewpoint, behavioral norms impose constraints to their actions in order to maintain a social order. Conversely, from an organizational perspective it is useful to separate the static aspect from the dynamic one, in this paper this is done by respectively introducing Structural and Adaptive Norms. The formers define the static structural aspect of the system at design time and provide the initial composition required to the organization to fulfill its objective. The latters govern the state transition of the organization from a given configuration to a new configuration to fit the environmental changes.

The proposed metamodel fully supports, and we experimented that by developing the IMPULSO system, a methodological approach for holonic multi-agent system design in which the holons are ruled by means of norms. In the future we will improve the design process based on the new metamodel that is obviously an extension of ASPECS.

References

1. Abran, A., Moore, J., Bourque, P., Dupuis, R., Tripp, L.: SWEBOK®: Guide to the Software Engineering Body of Knowledge. IEEE Computer Society (2004)
2. Bellifemine, F., Poggi, A., Rimassa, G.: JADE–A FIPA-compliant agent framework. In: Proceedings of PAAM, London, vol. 99, pp. 97–108 (1999)
3. Boella, G., Van Der Torre, L., Verhagen, H.: Introduction to normative multiagent systems. Computational & Mathematical Organization Theory 12(2), 71–79 (2006)
4. Boissier, O., Bordini, R., Hübner, J., Ricci, A., Santi, A.: Multi-agent oriented programming with jacamo. Science of Computer Programming (2011)
5. Bordini, R., Hubner, J., Wooldridge, M.: Programming multi-agent systems in AgentSpeak using Jason. Wiley-Interscience(2007)
6. Cossentino, M., Gaud, N., Hilaire, V., Galland, S., Koukam, A.: ASPECS: an agent-oriented software process for engineering complex systems. Autonomous Agents and Multi-Agent Systems 20(2), 260–304 (2010)
7. Dignum, V., Dignum, F.: Modelling agent societies: Co-ordination frameworks and institutions. In: Brazdil, P.B., Jorge, A.M. (eds.) EPIA 2001. LNCS (LNAI), vol. 2258, pp. 191–204. Springer, Heidelberg (2001)
8. Dignum, V., Meyer, J., Weigand, H., Dignum, F.: An organization-oriented model for agent societies. In: Proceedings of International Workshop on Regulated Agent-Based Social Systems: Theories and Applications (2002)
9. Ferber, J.: Multi-agent systems: an introduction to distributed artificial intelligence, vol. 222. Addison-Wesley, New York (1999)
10. Fischer, K., Schillo, M., Siekmann, J.: Holonic multiagent systems: A foundation for the organisation of multiagent systems. In: Mařík, V., McFarlane, D.C., Valckenaers, P. (eds.) HoloMAS 2003. LNCS (LNAI), vol. 2744, pp. 71–80. Springer, Heidelberg (2003)
11. Gerber, C., Siekmann, J., Vierke, G.: Holonic multi-agent systems. Technical report, Universitè- und Landesbibliothek (1999)
12. Hannoun, M., Boissier, O., Sichman, J.S., Sayettat, C.: MOISE: An organizational model for multi-agent systems. In: Monard, M.C., Sichman, J.S. (eds.) SBIA 2000 and IBERAMIA 2000. LNCS (LNAI), vol. 1952, pp. 156–165. Springer, Heidelberg (2000)
13. Hubner, J., Sichman, J., Boissier, O.: Moise+: towards a structural, functional, and deontic model for mas organization. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1, p. 502. ACM (2002)
14. Hubner, J., Sichman, J., Boissier, O.: Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. International Journal of Agent-Oriented Software Engineering 1(3), 370–395 (2007)
15. Isern, D., Sánchez, D., Moreno, A.: Organizational structures supported by agent-oriented methodologies. J. Syst. Softw. 84, 169–184 (2011)
16. Koestler, A.: The ghost in the machine. Psychiatric Communications 10(2), 45 (1968)

17. Mella, P.: The holonic revolution: Holons, holarchies and holonic networks: The ghost in the production machine (2009)
18. Morreale, V., Bonura, S., Francaviglia, G., Centineo, F., Cossentino, M., Gaglio, S.: Reasoning about goals in BDI agents: the PRACTIONIST framework. In: Proceedings of the 7th WOA 2006 Workshop, From Objects to Agents (Dagli Oggetti Agli Agenti). CEUR Workshop Proceedings, vol. 204, pp. 26–27. CEUR-WS.org (2006)
19. Omicini, A., Ricci, A., Viroli, M.: Artifacts in the A&A meta-model for multi-agent systems. Autonomous Agents and Multi-Agent Systems 17(3), 432–456 (2008)
20. Pokahr, A., Braubach, L., Lamersdorf, W.: Jadex: A BDI reasoning engine. In: Multi-Agent Programming, pp. 149–174 (2005)
21. Rao, A.: AgentSpeak (L): BDI agents speak out in a logical computable language. In: Perram, J., Van de Velde, W. (eds.) MAAMAW 1996. LNCS, vol. 1038, pp. 42–55. Springer, Heidelberg (1996)
22. Ricci, A., Viroli, M., Omicini, A.: cArtAgO: A framework for prototyping artifact-based environments in MAS. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2006. LNCS (LNAI), vol. 4389, pp. 67–86. Springer, Heidelberg (2007)
23. Therborn, G.: Back to norms! On the scope and dynamics of norms and normative action. Current Sociology 50(6), 863 (2002)
24. Toledo, C.M., Bordini, R.H., Chiotti, O., Galli, M.R.: Developing a knowledge management multi-agent system using *JaCaMo*. In: Dennis, L., Boissier, O., Bordini, R.H. (eds.) ProMAS 2011. LNCS, vol. 7217, pp. 41–57. Springer, Heidelberg (2012)
25. Ulieru, M., Brennan, R., Walker, S.: The holonic enterprise: a model for Internet-enabled global manufacturing supply chain and workflow management. Integrated Manufacturing Systems 13(8), 538–550 (2002)
26. Vázquez-Salceda, J., Dignum, V., Dignum, F.: Organizing Multiagent Systems. Journal of Autonomous Agents and Multi-Agent Systems 11(3), 307–360 (2005)
27. Winikoff, M.: Jack^{TM} intelligent agents: An industrial strength platform. In: Multi-Agent Programming, pp. 175–193 (2005)
28. Wooldridge, M.: Reasoning about rational agents. The MIT Press (2000)

Specification of Trade-Off Strategies for Agents: A Model-Driven Approach

René Schumann[1], Zijad Kurtanovic[2], and Ingo J. Timm[3]

[1] HES-SO,
Rue de Technopôle 3, 3960 Sierre, Switzerland
rene.schumann@hevs.ch
[2] University of Hamburg
22527 Hamburg, Germany
kurtanovic@informatik.uni-hamburg.de
[3] University of Trier
54286 Trier, Germany
ingo.timm@uni-trier.de

Abstract. Negotiation is an important part of today's business processes on an inter-enterprise level. Agent research offers a variety of approaches and tools to automate negotiations. Currently these technologies have the drawback that the human manager retains the responsibility for the outcome of a negotiation, but this person most often does not have the required knowledge to define the agent's behavior by himself. To increase acceptability of automated negotiation approaches, we consider it necessary that human negotiators can specify the strategies for the agents. In this article we present a meta model, which enables human negotiators to specify trade-off strategies. Trade-off strategies are a key concept in negotiation in general. This meta model is based on the Ecore meta model. The specified trade-off strategies can automatically be transformed into representations that can be used by an agent. Our meta model provides a model and a graphical notation that allows it to create graphical editors for trade-off strategies. Therefore, it becomes possible to specify trade-off strategies without any programming knowledge.

Keywords: automated negotiation, trade-off strategies, MDD.

1 Introduction

Negotiation is an important part of today's business processes on an inter-enterprise level. Agent research offers a variety of approaches and tools to automate negotiations. To enable agents to act on behalf of humans, several challenges have to be dealt with. From our perspective we see a particular challenge in the fact, that the human principal of an agent retains responsibility for the outcome of an automated negotiation. Thus, to establish agent-based negotiations it is required to a) give quality guarantees to the human principal, or b) allow the principal to specify the negotiation strategy of the agent. Independently of the chosen approach it has to be discussed if humans are willing to delegate negotiation tasks to agents.

Also it seems reasonable that automated negotiation can be support for humans, and that the human negotiator has in the end to accept the proposed deal of the system, before it becomes legally binding. In this article we restrict ourself to discuss the technique how negotiations can be realized, that follow a given negotiation strategy. How these techniques afterwards can be integrated into systems designed to support human negotiators is beyond the scope of this paper.

In real-world negotiation, multiple parameters as well as non-rational behavior of actors have to be considered. Thus, formal guarantees, i.e., proof of optimal behavior does not seem to be feasible. Therefore, we are focusing here on the second approach, enabling the human principal to specify her strategy. Such an approach includes means for the acquisition of negotiation knowledge from the human principal [1] of the agent, typically this is a manager or a negotiator. The negotiator has implicit knowledge about the negotiation process and related negotiation strategies. The knowledge on the process will be encoded in protocols. The knowledge on strategies is essential to automated negotiation in practical settings and has to be encoded, too. In this article we focus on trade-off strategies. In particular we present a meta model for trade-off strategies, and how strategies that have been defined based on this model, can be automatically transformed into a format that can be used by a negotiating agent.

A trade-off between two negotiation attributes specifies a preference among pairs of assignments to both variables. The idea of a trade-off is to improve one attribute while worsening the other in return [1]. Trade-offs are an important aspect of negotiations in human behavior [2,3] and have been adopted for negotiations among software agents, see e.g. [4,1].

A current problem in the application of autonomous agents as negotiators, or agents that support human negotiators, is that the human negotiator is not capable of designing or programming the agent, even though he remains responsible for the outcome of the negotiation. While on the other side the programmers can have problems understanding the strategy used during the negotiation.s In our research we want to provide means to bridge this mismatch, by providing means for specification of trade-off strategies that enables human negotiators to specify their trade-off strategies in a comprehensive way. Of course, trade-off strategies are only one part of negotiation strategies, which comprises also other aspects, like the protocols or the effects of time passing during the negotiation, e.g. if a deal has to be reached within a given time.

We use a model-driven development (MDD) approach to automatically translate a specification of a trade-off strategy into a representation that can be used by a software agent. Thus, the person specifying the strategies is not required to have knowledge about software agents or programming. Consequently, we empower the person responsible for the negotiation, the principal of the agent, to specify the strategy of the agents by himself. The need for encoding these strategies by hand becomes obsolete.

The rest of this article is structured as follows. In the next section we outline related work. Afterwards we describe formally the concept of trade-off strategies (Section 3.1). Based on this background we present our meta model (Section 3.2) and how it can be used to specify a trade-off strategy (Section 3.3). The transformation process of a strategy is outlined in Section 3.4. In Section 4 we present an example how our meta model can be used to specify trade-off strategies in a show case. Finally, we summarize our work and outline future research.

2 Related Work

In this paper, we focus on means of modeling trade-off strategies to enable humans to specify their strategies with the goal to support them by (partially) automate the negotiation process.

Automated negotiation is an established and active research topic in multiagent systems research, see e.g. [5,6]. To increase user acceptance, agents which act on behalf of humans in negotiations require knowledge from the human expert. Surprisingly, it has to be stated that only little research has been done in investigating how to acquire negotiations strategies from human experts in the field of automated negotiations, the work by Lou et al. [1] is one of those rare exceptions. Only very few work has been done to provide means that would allow to specify negotiation strategies in a declarative way.

One exception is the work by Chiu et al. [7]. The author present an e-Negotiation process based on an ontology. The process should support human negotiators to specify their negotiation strategies. The improvement of our work is, that we have combined means to specify trade-off strategies with MDD techniques to automatically generate a representation that can be used by agents, without any additional human effort.

Benyoucef and Rinderle [8] have presented a model-driven approach for developing service-oriented negotiation systems. Their specification of the negotiation behavior uses also a declarative approach for the specification. In our paper we strictly focus only to trade-off strategies.

In [1] user's trade-off strategies and preferences are acquired by using the *default-then-adjust* method. This approach is based on the use of a preexisting default knowledge with the aim to assist the user and reduce their workload. On the one side, such knowledge can be an important assistance for the user, on the other side the access to relevant expertise is limited and often not available [9].

Our work is based on the definition of trade-off- and preference-functions presented in [1], we have modified them slightly as follows:

– In [1] it is assumed that the domains of negotiation attributes are all continuous and numeric. We have relaxed this assumption and can handle domains with symbolic values and also attributes with discrete domains, too. This is done by mapping symbolic values to numeric values.
– To avoid formal case-based considerations and to be able to base a trade-off strategy on a pair of attributes with heterogeneous value sets, the domains of all negotiation attributes are represented as numeric and continuous. In the agent's reasoning his proposals are based on this assumption and are then approximated to become conform with the actual domains.

Another field in multiagent system research that is considered as related work here is the application of MDD techniques to multiagent system development. The probably most widely known system for the model-driven development of multiagent systems (MAS) is the INGENIAS system [10], which can be considered as a forerunner for MDD development of MAS. Recently, Hahn et al. have developed a coherent modeling framework for MAS. In their effort Hahn et al. have developed a platform-independent meta model for MAS [11], and afterwards detailed their model by adding modeling capabilities for interaction protocols [12]. In current research efforts the development of MDD tooling for MAS is top-down. Meta models are presented for MAS in general, as pointed out above. Their goal is to allow the modeling of entire MAS and their generation. In the approach we present here we only cover one aspect, namely the modeling of trade-off strategies. This bottom-up approach tries to come up with dedicated aspects of a MAS, that can be put in use early on. Also it would be interesting to see to what extend the work proposed in this article can be used in existing approaches. The negotiation meta model is not designed to be exclusive to other approaches, but allows for a more detailed modeling of a specific aspect of a MAS.

3 A Meta Model for Trade-Off Strategies

Before we are going to present the meta model for trade-off strategies (Section 3.2) we provide a brief definition of the concepts used. Afterwards, we detail how a trade-off strategy can be specified (Section 3.3) and transformed (Section 3.4) to be used by an agent during a negotiation.

3.1 Trade-Off Strategies

A trade-off strategy specifies what combination of attributes' values form an acceptable deal for the user. Within a trade-off strategy all information about the trade-offs is encoded. Formally, a trade-off strategy contains a set of trade-off relations and a set of *independent* attributes. Independent attributes are not member of a trade-off relation with other attributes.

A trade-off is a relation between two negotiation attributes. It defines that in favor for worsening one attribute the other one has to improve. Note that we focus here on binary trade-off relationships, since these are the most common ones [3]. According to [1] a trade-off function can be formalized as follows:

Definition 1
Let the domain of the attribute x be $X = [l_x, r_x]$, and the domain of y be $Y = [l_y, r_y]$. Note the domains values are ordered. A function $f : X \rightarrow Y$ is a trade-off function if it is continuous, monotonic *and met the* boundary condition. *The boundary condition ensures, that if one attribute is assigned to the best value, the other attribute has to be made worse [1].*

If a trade-off function exists between two attributes they are also called to be a trade-off pair. For each trade-off a preference function $p : A \times B \rightarrow [0, 1]$ is defined, which define the preference over the trade-off alternatives. It reflects a trapezoid formula of three segments (analogue to the preference function in [1]) to describe the increasing, steady and decreasing preference over trade-off alternatives. Trade-off and preference functions can also be specified graphically. An example is shown in Figure 1. In Figure 1 right hand side, a trade-off function between price (on the x-axis) and accuracy (on the y axis) is shown. A preference function indicates the degree of satisfaction (Figure 1 left hand side) of the negotiator, expressed in terms of an interval between [0,1]. Thereby, 1 is indicating high satisfaction and 0 is indicating dissatisfaction.

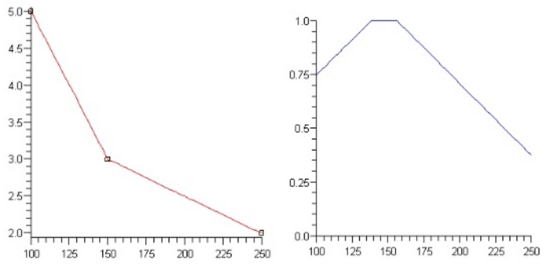


Fig. 1. Left: Example of a trade-off function between price (x-axis) and accuracy (y-axis), Right: Preference function about prices. Prices x-axis, Satisfaction degree y-axis.

Preference functions are also defined for independent attributes as $p : A \rightarrow [0, 1]$, with A: $\forall a, b \in A : \; a \preceq b \Leftrightarrow p(a) \leq p(b)$.

Following the previous definition a trade-off strategy can be represented as a forest, as illustrated in Figure 2: the nodes represent negotiation attributes and the edges trade-off relations.

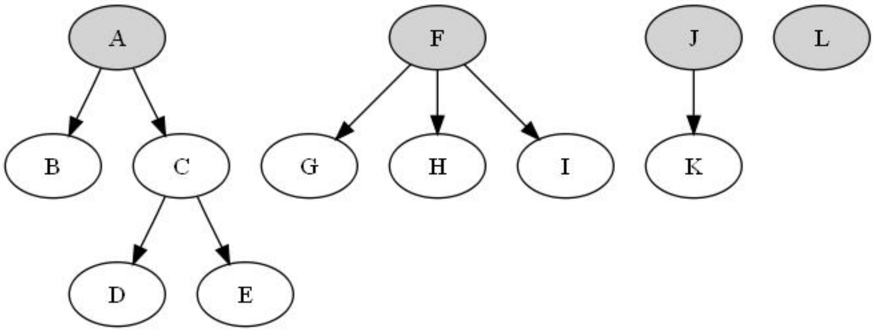


Fig. 2. Representing a trade-off strategy as a forest. L is an independent attribute.

This ensures some formal but also informal benefits. A trade-off strategy can be visualized in a *clear* and *accustomed* way to the users. Due to the acyclic structure there cannot exist inconsistencies, which may be introduced by cycles. This reduces the complexity for specifying and validating these strategies. Moreover, the forest structure allows to use of efficient algorithms for reasoning about the trade-off strategies [13].

3.2 Meta Model

We base our *negostrategy*-meta model on the meta model Ecore of the Eclipse Modeling Framework (EMF)[1]. As shown in Figure 3 the meta model for trade-off strategies has been modeled in two packages. Within the `base` package

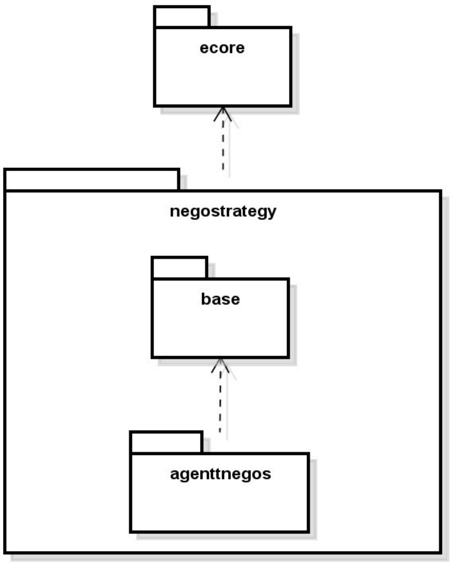


Fig. 3. Package diagram of the trade-off meta model

`negotiation attributes` are defined that can be used to define trade-off strategies. Also basic operators and other term are defined in the `base` package. The `base` package is detailed in Figure 4.

Trade-off relations are defined in the `agentnegos` package (Figure 5). Within this package we distinct between attributes that describe the context, in which a trade-off relation is valid, and attributes used within a trade-off relation. A trade-off strategy is formed by a set of trade-off relations and their context. The context allows to specify when a particular relation is applicable.

[1] see `http://www.eclipse.org/modeling/emf`, accessed at 27.10.2012.

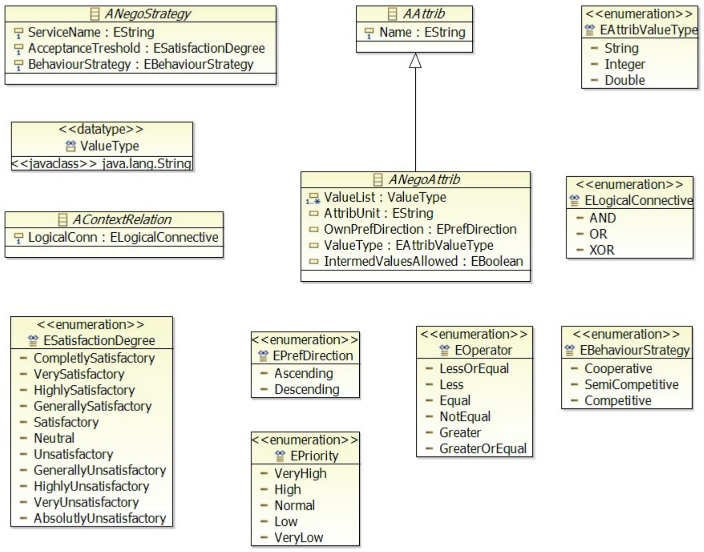


Fig. 4. The `base` package of the trade-off strategy meta model

In the following we highlight the three major concepts of the proposed meta model in more details, these are *negotiation strategies root*, *trade-off negotiation attribute*, and *trade-off relation*. The entire meta model can be found in [14]. For most of the concepts we have also defined a graphical notation, to allow further extensions, like a graphical editor for defining trade-off strategies, following our vision that the agent behavior can be defined by the principal of the agent, and not by a programmer.

The *negotiation strategy root* contains all negotiation attributes and their trade-off relationships. It can be seen as the artificial root node, for the entire forest representing a trade-off strategy. Each tree in the trade-off strategy has a priority, so the relative value between the trees can be encoded. Also an acceptance threshold is stored in the strategy. The acceptance threshold specifies a value that the agent use to a) generate an offer that is acceptable or b) decide on acceptability of an offer for the user. The graphical representation for this concept is a trapezoid containing the name of the service, as shown in Figure 6(a).

A *trade-off negotiation attribute* represents a negotiation attribute used in a trade-off strategy. It has a name, a domain, which can be an continuous interval([]) or a discrete enumeration({}), and a preference function over it's domain, plus a list of trade-off relationships in which this attribute is involved. The graphical representation is shown in Figure 6(b).

A *trade-off relation* encodes the trade-off function between two attributes. A relation defines the optimal combination of values between the two attributes. Given the optimal combination between the attribute values of the trade-off function and the preference functions, the trade-off combinations can be computed and ranked. A *trade-off relation* is represented as a labeled edge connecting the

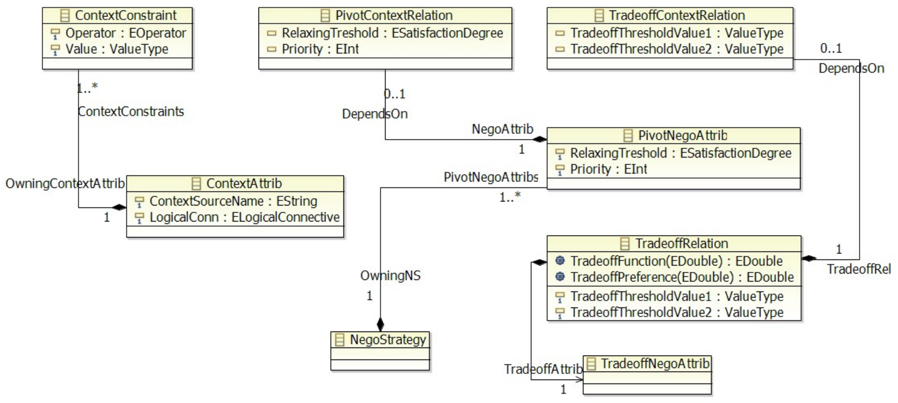
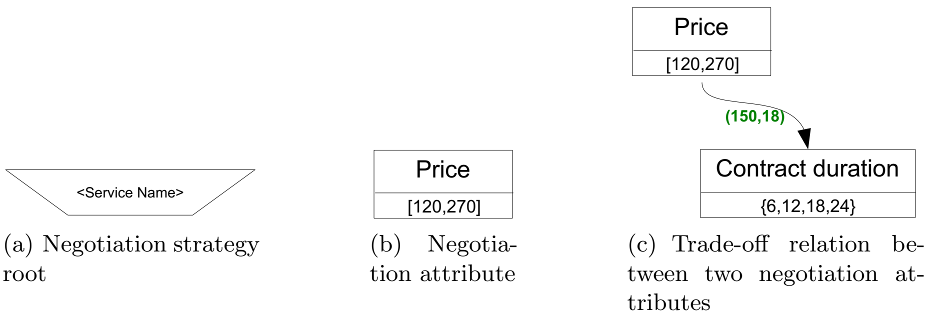

ContextConstraint
Operator : EOperator
Value : ValueType
PivotContextRelation
RelaxingTreshold : ESatisfactionDegree
Priority : EInt
TradeoffContextRelation
TradeoffThresholdValue1 : ValueType
TradeoffThresholdValue2 : ValueType
0..1
DependsOn
1..*
ContextConstraints
0..1
DependsOn
NegoAttrib
1
PivotNegoAttrib
RelaxingTreshold : ESatisfactionDegree
Priority : EInt
OwningContextAttrib
ContextAttrib
ContextSourceName : EString
LogicalConn : ELogicalConnective
1
PivotNegoAttribs
1..*
TradeoffRelation
TradeoffFunction(EDouble) : EDouble
TradeoffPreference(EDouble) : EDouble
TradeoffThresholdValue1 : ValueType
TradeoffThresholdValue2 : ValueType
1
TradeoffRel
OwningNS
1
NegoStrategy
TradeoffAttrib
1
TradeoffNegoAttrib

Fig. 5. The `agentnegos` package of the trade-off strategy meta model



(a) Negotiation strategy root

(b) Negotiation attribute

(c) Trade-off relation between two negotiation attributes

Fig. 6. Graphical representation for negotiation strategy concepts

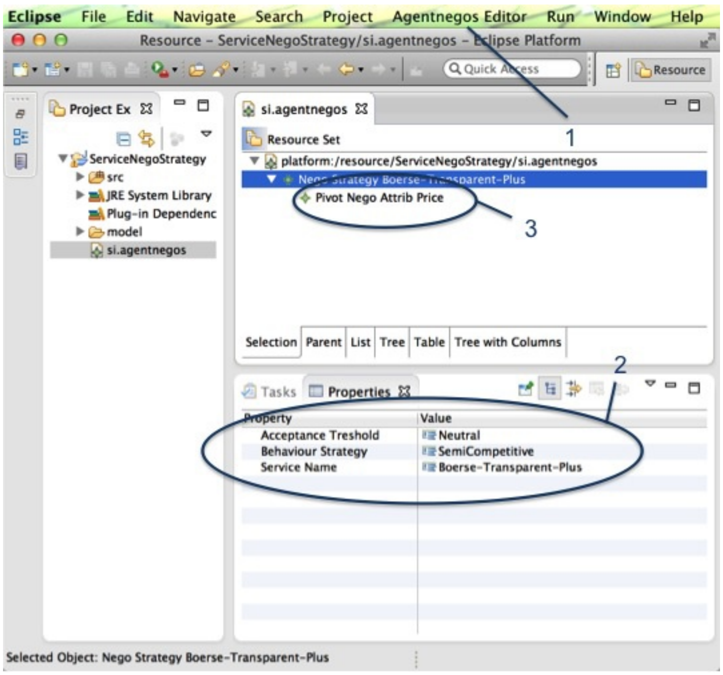
two negotiation attributes. The label is the optimal value combination of both attributes. An example is shown in Figure 6(c).

Additionally, it is possible to specify the context for which the trade-off relations are defined. For instance, it is possible to differentiate the strategies depending on with whom the agent negotiate.

3.3 Specification of Trade-off Strategies

As we use the EMF as a base for our meta model we have the option to provide a graphical editor for negotiation strategies in the near future. As outline above, this would enable strategy specification by non IT-experts, which most often comprises the persons responsible for the outcomes of a negotiation, e.g. a manager.

Currently, a negotiation strategy is specified using an Eclipse widget that has been generated automatically based on the specification of the meta model.


Eclipse File Edit Navigate Search Project Agentnegos Editor Run Window Help
Resource - ServiceNegoStrategy/si.agentnegos - Eclipse Platform
Quick Access
Resource
Project Ex
ServiceNegoStrategy
src
JRE System Library
Plug-in Dependenc
model
si.agentnegos
si.agentnegos
Resource Set
platform:/resource/ServiceNegoStrategy/si.agentnegos
Nego Strategy Boerse-Transparent-Plus
Pivot Nego Attrib Price
Selection Parent List Tree Table Tree with Columns
Tasks Properties
1
2
3

Property	Value
Acceptance Treshold	Neutral
Behaviour Strategy	SemiCompetitive
Service Name	Boerse-Transparent-Plus

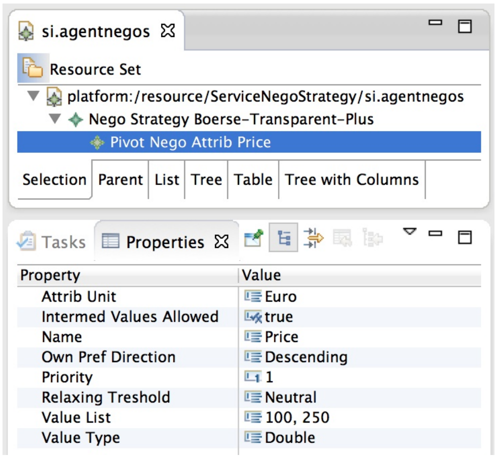
Selected Object: Nego Strategy Boerse-Transparent-Plus

Fig. 7. Overview of the current trade-off strategy editor: 1) points at the particular tab for editing the trade-off strategy 2) panel for the editing of attribute values 3) panel for the navigation within the attribute tree forming the data structure used for the trade-off strategy

A screen shot is shown in Figure 7. Of course this form is not suitable for non IT experts. Therefore a graphical editor is needed. The strategy is specified in form of structured attribute value pairs, as shown in Figure 8. The specified strategy will be validated against the meta model and saved in the XMI format.

3.4 Model Transformation

If a trade-off strategy has been specified it needs to be transformed into a representation that can be used by a software agent. This requires platform-specific details which supplies the EMF-generator with information like the connections between multiple Ecore-models, the name of the generated files, referenced Ecore-models etc. [15]. Based on the XMI file a generator transforms a trade-off strategy into a representation that can be used by an agent. We have decided to use a relational representation. In this representation all acceptable deals, i.e., combination of values for negotiation attributes that exceed the specified acceptance threshold, are stored.



Property	Value
Attrib Unit	Euro
Intermed Values Allowed	true
Name	Price
Own Pref Direction	Descending
Priority	1
Relaxing Treshold	Neutral
Value List	100, 250
Value Type	Double

Fig. 8. Detailed view on the panel for editing the negotiation strategy

These deals are precomputed, because the problem of finding the next best offer in a negotiation process would be too time consuming, for details see Lou et al. [4]. In the same way the set of acceptable deals is precomputed. This shifts the computational efforts form the execution into the compilation phase. During the negotiation, queries are performed to retrieve elements with specific characteristics in the set of acceptable deals. This querying can be done efficiently using a relational representation, e.g., in form of SQL queries [16].

A trade-off strategy is transformed into a set of tables. For each tree of a trade-off strategy a table is generated. Each negotiation attribute is represented by a column. Each row specifies one acceptable deal. The set of tables is computed by the function `generateTables` presented in Algorithm 1. For each tree of a trade-off strategy, first a representation set of the root is generated. Afterwards the representation sets of the direct child-nodes are induced by the corresponding trade-off functions. Each directed edge in a tree represents a trade-off relation from *parent* node to *child* node.

Consequently, a resulting table has a column for the parent node's values, possible several columns holding child node's values, a column with the preference values and one column with preference values with priority degree considered. The structure of such a table is outlined in Table 1. In the resulting table preferences and priority-preferences are aggregated, e.g. using the arithmetic mean. Examples for such tables can be seen below in Figures 12 and 13. For retrieving information about a negotiation attribute it can become necessary to join several tables.

Table 1. Example of a table representing a trade-off relation between *ParentAttrib1* and *ChildAttrib1*

ParentAttrib1	ChildAttrib1	Pref	PrioPref
...	...	...	...

Algorithm 1. Pseudo code of the tree transformation algorithm

```
function GENERATETABLES(set of trees forming a trade-off strategy)
    for each Tree do
        generate a representation set of the root
        Call InduceRepresentationSet(root)
    end for
end function
function INDUCEREPRESENTATIONSET(Node X)
    if X NOT ROOT then
        Induce a representation set from X
    end if
    for each Child-Node C of X do
        InduceRepresentationSet(C)
    end for
end function
```

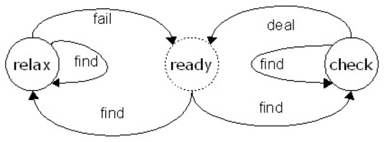
4 Automating Negotiations: A Case Study

In this section we demonstrate the specification of a negotiation by an example. We show for a simple negotiation scenario how trade-off specification can be done, and the resulting negotiation outcomes.

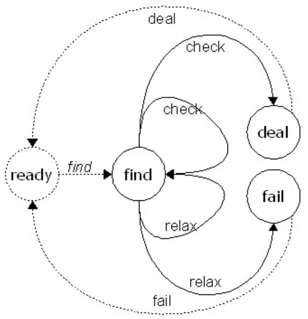
4.1 Negotiation Model

Since we focus here on the trade-off strategies, we choose an existing negotiation model presented by Lou et al. [4]. It is a simple bilateral negotiation setting. Two roles are defined: a buyer and a seller agent. Both agents negotiate about a contract with a number of attributes like price, quality, delivery or payment date. Each agent has a global preference function for all permutations of all possible outcomes of the negotiation. The seller provides access to an information service, that the buyer wants to subscribe to. Attributes of the contract are price, actuality of the data, contract duration, and accounting period.

The agents operate in a semi-competitive environment. This is reflected by their behavior strategies which are based on the *principled negotiation approach* [2]. They try to weaken their position only minimally, e.g., by minimal information disclosure, and minimal relaxation of their desires [4]. The negotiation protocol is based on the alternating offers protocol [17]. The behavior of the seller agent is presented in Figure 9(a). The `ready` state is the initial state. The states `check` and `relax` represent the allowed performatives of sending messages of an agent in a given negotiation state. The edges represent the performatives of the buyer agent that can be received during a negotiation encounter. When the performative *find* is received the negotiation is initiated and the agent can answer with the performatives *check* or *relax*. The performative *check* is used to ask the other agent to check if an offer satisfies its requirements. If no offer could be found that satisfies the published buyers constraints, the seller asks to *relax* at least one of the constraints, so that a suitable offer can be found.



(a) Seller agent's behavior protocol



(b) Buyer agent's behavior protocol

Fig. 9. Negotiation protocol for seller and buyer

The buyer agent's behavior protocol is presented in Figure 9(b). The states and edges are defined analogous to figure 9(a). The buyer starts a negotiations by sending a *find*-performative to the seller agent. Agents constrains future offers by sending constraints that all offers have to fulfill, e.g., the price should be below 340€. Constraints are published with descending priority. An offer is checked and either accepted or another constraint is published to specify the requirements more precisely, by communicating the violated constraint with the highest priority. The constraints that needs to be satisfied and the preference function among all available feature combination of negotiation attributes encodes the negotiation strategy. Both have to be defined by the principal of the agent.

4.2 Specification of Trade-off Strategies for the Example

From the seller's perspective the negotiation attributes have the following domains: The price can be in a range between [120,270] €, of course a higher price is preferred. The delivered data can have an actuality of 1,2,4 or 6 hours. As more accurate data is more expensive, older data is preferred. The seller assumes its optimal ration between profit and accuracy is€ 170,- for two hour old data. Possible contract durations are 6,12,18 or 24 month, a longer duration is preferred. Accounting periods are 1,3,4 or 6 months, shorter periods are preferred, not giving a credit to the customer. Based on the notation presented in Section 3 the resulting trade-off strategy can be modeled as shown in Figure 10. v

From the buyer's perspective the attributes have other desired values and preferences, of course. The price should be in the interval between [100,200] €, and a lower price is preferred. Actuality of the data should be between two and five hours, more accurate data is preferred and a higher price is acceptable. A fair ratio between accuracy and price for the buyer is paying € 150,- for three hours old data. The contract duration can be in the interval between [3,24] month, where a shorter duration is preferred. For a better (for the buyer a lower) price

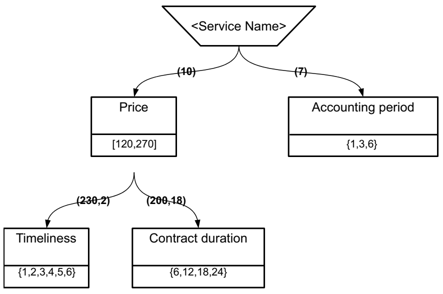


Fig. 10. Graphical representation of the seller trade-off strategy

the buyer is willing to accept a longer contract duration. Acceptable accounting periods can be one to three month. Longer periods are preferred, but for a better price, shorter periods can be accepted. The graphical model for this strategy is shown in Figure 11.

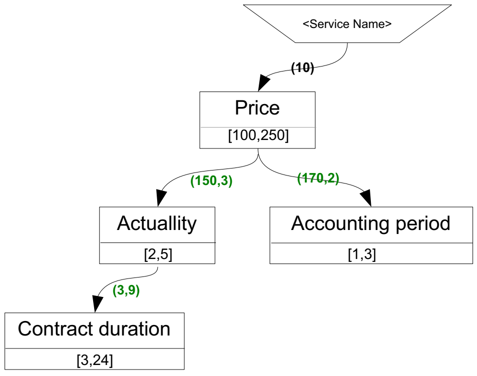
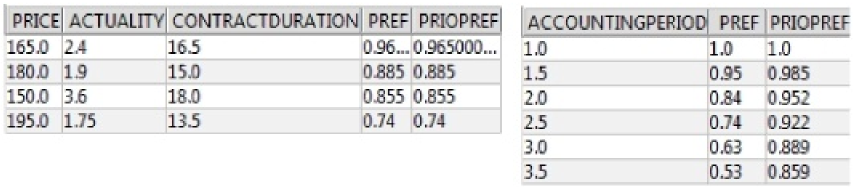


Fig. 11. Graphical representation of the buyers trade-off strategy

We have specified these two strategies with our editor and generated the relational representation for these strategies. The resulting set of acceptable deals have been generated. The relational representation for the seller's strategy is shown in Figure 12 and for the buyer's strategy in Figure 13.

We have implemented negotiating agents, based on the Jade framework[2], that use the relation representation to negotiate with each other. In Table 2 we present the negotiation process, as it has been executed by the agents for the described example. The buyer starts the negotiation by selecting the row of the table shown

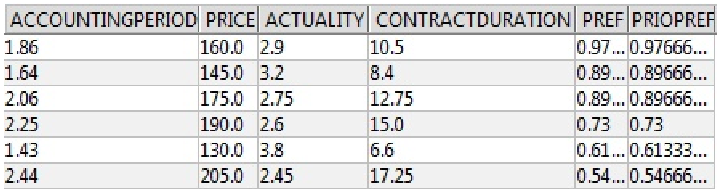
[2] see `http://jade.tilab.com`, accessed at 27.10.2012.



PRICE	ACTUALITY	CONTRACTDURATION	PREF	PRIOPREF
165.0	2.4	16.5	0.96...	0.965000...
180.0	1.9	15.0	0.885	0.885
150.0	3.6	18.0	0.855	0.855
195.0	1.75	13.5	0.74	0.74

ACCOUNTINGPERIOD	PREF	PRIOPREF
1.0	1.0	1.0
1.5	0.95	0.985
2.0	0.84	0.952
2.5	0.74	0.922
3.0	0.63	0.889
3.5	0.53	0.859

Fig. 12. Relational representation of seller's trade-off strategy. His thresholds are already considered, thus only acceptable trade-offs are shown.



ACCOUNTINGPERIOD	PRICE	ACTUALITY	CONTRACTDURATION	PREF	PRIOPREF
1.86	160.0	2.9	10.5	0.97...	0.97666...
1.64	145.0	3.2	8.4	0.89...	0.89666...
2.06	175.0	2.75	12.75	0.89...	0.89666...
2.25	190.0	2.6	15.0	0.73	0.73
1.43	130.0	3.8	6.6	0.61...	0.61333...
2.44	205.0	2.45	17.25	0.54...	0.54666...

Fig. 13. Relational representation of all possible attribute combinations of the attributes price, actuality and contract duration

Table 2. Full negotiation trace of buyer and seller (PR: price, AC actuality, CD contract duration, AP accounting period)

Round 1	Buyer	Performative: Find Constraint: PR $\leq$ 160
	Seller	Performative: Check (PR:150,AC:4,CD:18,AP:1)
Round 2	Buyer	Performative: Find Constraint: PR $\leq$ 160 $\wedge$ AC $\leq$3
	Seller	Performative: Relax
Round 3	Buyer	Performative: Find Constraint: PR $\leq$ 145 $\wedge$ AC $\leq$3
	Seller	Performative: Relax
Round 4	Buyer	Performative: Find Constraint: PR $\leq$ 175 $\wedge$ AC $\leq$3
	Seller	Performative: Check (PR:165,AC:2,CD:18,AP:1)
Round 5	Buyer	Performative: Find Constraint: PR $\leq$ 175 $\wedge$ AC $\leq$3 $\wedge$ CD $\leq$ 13
	Seller	Performative: Relax
Round 6	Buyer	Performative: Find Constraint: PR $\leq$ 190 $\wedge$ AC $\leq$3 $\wedge$ CD $\leq$ 15
	Seller	Performative: Check (PR:180,AC:2,CD:12,AP:1)
Round 7	Buyer	Performative: Find Constraint: PR $\leq$ 190 $\wedge$ AC $\leq$3 $\wedge$ CD $\leq$ 15 $\wedge$ AP $\geq$ 2
	Seller	Performative: Check (PR:180,AC:2,CD:12,AP:3)
Round 8	Buyer	Performative: Deal

in Figure 13 with the most preferred combination of attributes' values according to his trade-off strategy. According to his behavior strategy the agent tries to minimize the revelation of private information, thus revealing only one constraint per round to the seller, i.e. the agent requests a deal for a price $\leq$ €160,-. The seller then queries his possible deals to find a suitable offer. The seller sends it's most preferred bid to the buyer and asks him to evaluate it. In round 2 the buyer finds that the offer is not acceptable because some constraints are violated, e.g. for the offered price a better actuality of data and shorter contract duration is expected. In consequence the buyer asks the seller to find another offer satisfying the price and another published constraint, i.e. the actuality should equal or below 3 hours. The seller agent has no fitting offer and requests a relaxation of the constraints. In doing so the buyer lowers his expected satisfaction degree he will obtain in this negotiation. Finally, after 2 more unacceptable offers from the seller in rounds 4 and 6, a deal is reached in round 8.

5 Conclusion

In this article we presented a meta model, which enables human negotiators to specify trade-off strategies. As our meta model is based on the Ecore model, we were able to define code generators that transform trade-off models into a representation that can be used by software agents. With this approach it becomes possible that, e.g., a procurement manager can specify their trade-off strategies, and software agents can negotiate on their behalf. Following the MDD principle we can avoid the expensive and possibly erroneous process of encoding the negotiation strategies by hand. We have demonstrated the feasibility of our approach in an prototype capable to perform simple negotiations as shown in the previous section.

The vision of our research is to allow a human negotiator to specify their entire negotiation strategy in a form that can be transformed automatically into reasoning knowledge of an agent. Therefore the principal of the agent is not required to have any knowledge about software agents or programming. In the future we want to realize further steps towards this vision. We will extend our tooling to cover more aspects of negotiations. As our trade-off specification meta model includes a graphical notation, we are going to develop a visual editor for the specification of negotiation strategies, to making it more convenient for humans. Moreover, we want to automate more phases of the specification of software agent negotiations using MDD principles. So further steps can be the specification and automated transformation of negotiation protocols, including the embedding of the strategy specific behavior within the executable model of the protocol.

Another important non-technical aspect that needs to be covered, is to investigate under which conditions humans could be willing to completely automate negotiations, or are willing to accept propositions made by an automated negotiation system. Thus, after a sufficient tooling has been created it is necessary to study the acceptance of such a technology. This is also necessary to adapt the methodology and tooling towards a) scenarios in which a automation is accepted by the users, and b) towards the needs of the human negotiators that are willing to be supported by negotiating agents.

References

1. Luo, X., Jennings, N.R., Shadbolt, N.: Acquiring user tradeoff strategies and preferences for negotiating agents: A default-then-adjust method. Int. J. Hum.-Comput. Stud. 64(4), 304–321 (2006)
2. Fisher, R., Ury, W.: Getting to yes: Negotiating agreement without giving in, 2nd edn. Mifflin, Boston (1991)
3. Steele, P.T., Beasor, T.: Business negotiation: A practical workbook. Gower, Aldershot (1999)
4. Luo, X., Jennings, N.R., Shadbolt, N., Leung, H., Lee, J.: A fuzzy constraint based model for bilateral multi-issue negotiations in semi-competitive environments. Artificial Intelligence Journal 148(1-2), 53–102 (2003)
5. Rosenschein, J.S., Zlotkin, G.: Rules of Encounter: Designing Conventions for Automated Negotiation among Computers, 2nd edn. MIT Press, Cambridge (1998)
6. Sandholm, T.: Distributed rational decision making. In: Weiss, G. (ed.) Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, pp. 201–258. MIT Press (1999)
7. Chiu, D.K.W., Cheung, S.C., Hung, P.C.K.: Facilitating e-Negotiation Processes with Semantic Web Technologies. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, HICSS 2005, p. 36a. IEEE Computer Society (2005) ISSN=1530-1605, doi:10.1109/HICSS.2005.269
8. Benyoucef, M., Rinderle, S.: A model-driven approach for the rapid development of e-negotiation systems. In: EMISA, pp. 80–93 (2005)
9. Wikberg, P.: Eliciting Knowledge from Experts in Modeling of Complex Systems: Managing Variation and Interactions. PhD thesis, Linköping University, Department of Computer and Information Science (2007)
10. Pavón, J., Gómez-Sanz, J.: Agent oriented software engineering with INGENIAS. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS (LNAI), vol. 2691, pp. 394–403. Springer, Heidelberg (2003)
11. Hahn, C., Fischer, K.: A platform-independent metamodel for multiagent systems. International Journal on Autonomous Agents and Multi-Agent Systems (JAAMAS) 18(2), 239–266 (2009)
12. Hahn, C., Zinnikus, I., Warwas, S., Fischer, K.: Automatic generation of executable behavior: A protocol-driven approach. In: Gleizes, M.P., Gómez-Sanz, J.J. (eds.) AOSE 2009. LNCS, vol. 6038, pp. 110–124. Springer, Heidelberg (2011)
13. Russell, S.J., Norvig, P.: Artificial intelligence: A modern approach. [the intelligent agent book], 2nd edn. Prentice Hall series in artificial intelligence. Prentice Hall, Upper Saddle River (2003)
14. Kurtanovic, Z.: Spezifikation von Verhandlungsstrategien. Diploma thesis, Institute for Computer Science, Goethe University Frankfurt a.M. (2008)
15. Budinsky, F.: Eclipse modeling framework: A developer's guide. The eclipse series. Addison-Wesley, Boston (2003)
16. Apt, K.: Principles of Constraint Programming. Cambridge University Press, New York (2003)
17. Osborne, M.J., Rubinstein, A.: Bargaining and markets. Economic theory, econometrics, and mathematical economics. Acad. Press, San Diego (1990)

MDA-Based Approach for Implementing Secure Mobile Agent Systems

Slim Kallel, Monia Loulou, Molka Rekik, and Ahmed Hadj Kacem

ReDCAD Laboratory
FSEGS, University of Sfax, Tunisia
B.P. 1173, 3038 Sfax, Tunisia
{slim.kallel,monia.loulou,ahmed.hadjkacem}@fsegs.rnu.tn

Abstract. We propose an approach for implementing secure mobile agent systems. In the first step, we define a meta-model which extends the UML deployment diagram by concepts related to the security and mobility of multi-agent systems. We propose also a UML profile as an implementation of this meta-model. All defined concepts are based on formal specifications. In the second step, we project the application model in AGLETS-specific model, which describes the structure and the main functionalities of the application using the AGLETS concepts. In the third step, we generate Java skeleton code from the obtained model, and we generate also AspectJ code for enforcing security properties defined in the application model. The generated aspects will be woven, in modular way, into the functional application code.

1 Introduction

Security problems constitute a brake to the expansion of the mobile agent technology. In fact, security in mobile agent systems is twofold: it aims on the one hand, to protect the mobile agents and, on the other hand, to protect systems in which execute an incoming agent. Indeed, when an agent moves, it is crucial to ensure that it will be executed correctly on the new visited system. Similarly, it is crucial to reassure the agent system that there will not be any risk to receive a new mobile agent.

The design and the development of secure mobile agent-based applications must be rigorous and assisted in order to surmount the difficulties due to the complexity of basic concepts related to the mobility and the security. In this context, some research approaches have been proposed such as [1–3]. However several limitations have been identified:

First, the specification of security requirements is mostly limited to control the behavior of mobile agents and their resources access. *Second*, most of the proposed approaches do not apply an automatic process until the code generation. Moreover, these approaches do not consider the implementation on several mobile agents platforms; they take into account only one platform. *Third*, not all steps in the process of implementing security properties are covered; these properties are not specified in the early phases of software development but rather added later, which negatively affects the application code's quality. Therefore, these approaches do not support a refinement technique between the different phases until the code generation. *Fourth*, the implementation of security policies in mobile agent systems, often, poses serious problems: The code is not encapsulated in separate modules; it is mostly written and integrated manually with the functional application code. This lack of modularity leads to overlapping and interlacing the code. Thing which increases considerably the complexity of systems. In addition, there is no guarantee that the code implementing these requirements is conform to the specification.

To overcome all previous limitations, we propose a generic approach baptized MDS4MAS which combines the advantages of model-driven architecture and aspect-oriented programming for developing secure mobile agent systems. Our approach covers the whole development process of secure mobile agent systems: starting from the formal specification, the platform-independent modeling, the AGLETS specific modeling by automatic transformation, to the generation of functional application code and security code as aspects.

Our approach offers a better way to reduce the complexity of mobile agents and their needs for security. It supports also most of security concepts related to mobile agent systems. Based on aspect-oriented programming, our approach provides modular and maintainable security code. Finally, our approach can also be applied automatically in other multi-agent platforms based on the refinement techniques and the transformation rules for generating corresponding code.

The remainder of the paper is organized as follows: Section 2 presents the phases composing our approach. Section 3 gives an overview of the formal specification and verification of secure mobile agent systems and Section 4 presents the modeling of such systems. Section 5 presents the transformation of the application model into the AGLETS specific model. Section 6 describes the automatic code generation from the specific model. In Section 7, we illustrate our approach on an E-commerce case study. Section 8 reports on related work while Section 9 concludes the paper and presents areas of future work.

2 Our Proposed Approach

Our approach presumes a three-phase process as shown in Figure 1.

The first phase consists of designing with a high level of abstraction secure mobile agent systems. This phase corresponds to the PIM level of the MDA approach. We extended the UML deployment diagram by defining a new meta-model for specifying new concepts required to model the security and mobility of agent systems. We proposed also two UML profiles as an implementation of this meta-model. The first one *MobilityProfile* extends UML in order to model a mobile agent system, while the second profile *SecurityProfile* consists of modeling security aspects of such systems. In addition to a set of stereotypes, this profile defines a set of OCL constraints to specify security constraints. This phase is based on our previous work [4], which proposes a conceptual model that rigorously specifies the key concepts of secure mobile agent systems and unifies their

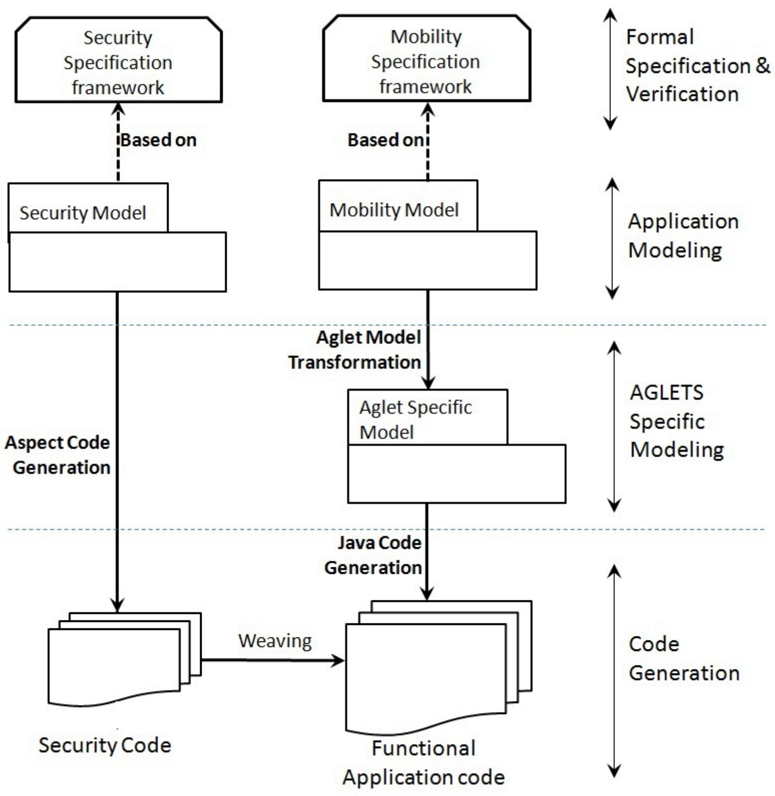


Fig. 1. An overview of MDS4MAS approach

representations independently of the specific application domain. Moreover, we presented a formal verification framework which gives more completeness and more consistency to the proposed specifications.

The second phase, known as PSM level in MDA approach, consists of generating a new model as a result of projecting the application model (described in the previous phase) in a specific platform. We defined and applied a set of transformation rules on the proposed model in order to generate automatically an AGLETS specific model. This model describes the structure and the main functionalities of the application using the AGLETS concepts.

In the third phase, we are interesting in the code generation process. Using Acceleo tool [5], we implemented a code generator based on two templates for generating the corresponding mobility and security code. The first template generates a Java skeleton code from the obtained AGLETS specific model (described in phase 2), while the second template consists of generating aspect code in AspectJ language from the OCL expression specifying the security constraints (described in phase 1). The generated aspect will be integrated in modular way into the functional application code for enforcing security constraints.

3 Formal Model for Secure Mobile Agent Systems

In this section, we present the formal specification of secure mobile agent systems [6] that support the expression of numerous security policy types in order to control the behavior of system entities and to protect them. In order to avoid any anomalies able to reduce the policy performance, we verify the consistency of the proposed specifications as well as the consistency intra-policy. All the proposed concepts are specified rigorously using Z notation [7] and checked using the Z/EVES toolkit [8].

3.1 Formal Specification

According to the study of several mobile systems, we present the most fundamental concepts of secure mobile agent systems. For mastering the complexity related to the mobility and the security of mobile agent systems, we adopt a high level of abstraction which eliminates any useless details in relation to the required properties.

A **Mobile Agent System** (MbAS) is a computer network composed of a set of interconnected *host machines*. Each one, has a unique name and described by a set of computing resources *CResource* which design its hardware features. On the same machine, one or more agents based systems *AgentSystem* can execute.

An **Agent System** (AgS) is composed of a set of agents (stationary/mobile) evolving within an environment. This latter offers the basic functionality for mobile agent execution. Indeed it ensures agent creation and initialization, reception of incoming agents, communication between agents, access to resources, agent migration, etc. These control services and others of application will be ensured by service agents.

In general, a **Mobile Agent** (MAg) is an active entity capable to migrate from one site to another in order to get nearer to the required resources and services to accomplish properly its goals. A MAg can be specified as a stationary agent which should have some others attributes to express its mobility.

Thus, a mobile agent should be identified by a name defined at its creation. This identity will be used to be able to localize the agent after its migration and communicate with it. In addition, a mobile agent must remember the site of its creation to be able to come back. The mobile agent acts according to its capability and its knowledge in order to achieve tasks that are affected to him. It defines its new localization according to its requirements in terms of resources and services and according to its partial view of inter-hosts connections. The sequence of visited systems constitutes the agent itinerary.

Both agent system and mobile agent should have well defined security policy with the aim to screen the incoming agents and/or adversary agent system respectively adversary mobile agent and hosting agent system. Thus, a secure entity *SEntity* can be either a mobile agent (*MAg*) or an agent system (*AgS*).

$$SEntity ::= MAg \langle\!\langle MobileAgent \rangle\!\rangle \mid AgS \langle\!\langle AgentSystem \rangle\!\rangle$$

Mobile agents and Agent systems aim to protect their secure objects denoted by *SObject*. A secure object may be either data *Data*, or service or computing resource *CResource* : : $SObject ::= D\langle\langle Data\rangle\rangle \mid Sr\langle\langle Service\rangle\rangle \mid Rs\langle\langle CResource\rangle\rangle$. A security policy *SecurityPolicy* regroups a set of security rules *SecurityRule*. Each one is described by :

– A type of the security rule *Type*. In order to express various kinds of security policies, *Type* may be either authorization or prohibition or obligation,
– The secure entity concerned with the security rule *Interested*. It may be either, a mobile agent or an agent system.
– The subject entity *RSubject* on which we apply the rule. It's defined by a non empty set of mobile agents or agent systems
– The target object *Target*. It defines the set of objects to be protected.
– A non empty set of actions *ControlledAction* to be enforced by the rule to reach the desired behavior.

We formally specified a security rule using Z notation as follows.

SRule

$Name : Propriety$
$Type : SConstruct$
$Interested : SEntity$
$RSubject : \mathbb{F}_1\ SEntity$
$Target : \mathbb{F}\ SObject$
$Context : Condition$
$Actions : \mathbb{F}_1\ Action$

$\forall r : CResource \mid Target = \{Rsr\} \bullet (Type = Auth \lor Type = Prohb)$ [C1]
$\quad \land (\exists s_1 : AgentSystem \bullet (Interested = AgSs_1 \land r \in s_1.Reserved\_res))$
$\quad\quad \land \neg (\exists s_2 : AgentSystem \bullet AgSs_2 \in RSubject)$
$\forall sc : Service \mid Target = \{Srsc\} \bullet (Type = Auth \lor Type = Prohb)$ [C2]
$\quad \land (\exists s_1 : AgentSystem \bullet (Interested = AgSs_1 \land sc \in s_1.Services))$
$\quad\quad \land \neg (\exists s_2 : AgentSystem \bullet AgSs_2 \in RSubject)$
$Type = Oblig \Rightarrow \{Context\} \neq \varnothing$ [C3]

The declarative part presents all security concepts, while the predicate part specifies the security related constraints.

The specification of a security rule must satisfy three constraints, given in the predicate part. For example [C1] states that when a target of a given rule is a computing resource, then the *Interested* entity in the rule must be an *AgS* and, indeed, the *RSubject* of the rule must be a *MAg*. Moreover, we check with [C1] that the AgS denoted with *Interested* can only control the access to its own resources.

Formally, a security policy is specified with the following schema:

$$
\begin{array}{|l}
\textit{SPolicy} \\
\hline
\textit{Subject} : \textit{SEntity} \\
\textit{Rules} : \mathbb{F}\, \textit{SRule} \\
\hline
\forall\, r : \textit{SRule} \mid r \in \textit{Rules} \bullet r.\textit{Interested} = \textit{Subject} \\
\forall\, a, b : \textit{SRule} \mid a \in \textit{Rules} \wedge b \in \textit{Rules} \wedge a \neq b \\
\qquad \bullet\ a.\textit{Name} \neq b.\textit{Name} \\
\hline
\end{array}
$$

In the predicate part, we check that a policy *SPolicy* regroups the security rules which have the subject defined in the declaration part. Moreover, we check that different rules have different names.

3.2 Formal Verification

Writing proofs is an essential part in order to show the consistency of the specification and consequently improves the quality of the desired software [7].

Conflicting and redundant security rules may reduce the performance of the policy and even make it inefficient. In fact, it is important to associate to the specification of security policies, a verification framework which checks the two main cases of policy inconsistencies : the modality conflicts and the redundancy of rules.

Regarding the adopted specification of security policies, we distinguish three different modalities which are authorization, prohibition and obligation. Two types of modality conflicts may occur :

– An authorized action is forbidden by a prohibition rule,
– An obligation rule may require to perform an action which is forbidden by a prohibition rule.

For modeling the relationships which may exist between two or several rules, three relations has been defined: an unary relation named *Consistent* and two binary relations named *Contradictory* and *Redundant.*

In order to prove the consistency of a given policy, we should check that there is no contradiction between the policy rules and there is no redundant rules. On that basis, a rigorous definition of policy consistency given by a rewriting-rule *Def_Consistent* is defined. It appeals the definition of *Redundant* and *Contradictory* relations. A complete description of the specification of *Contradictory* and *Redundant* relations is presented in [6]. To prove the consistency of a given policy requires to define a theorem which refers to the specification of the relation *Consistent.* Let's assume a security policy *Test_Policy*. To prove the consistency of *Test_Policy*, we add the following conjecture asserting:

$$
\begin{array}{|l}
Consistent\_ : \mathbb{P}\, SPolicy \\
\hline
\forall p : SPolicy \bullet Consistent\ p \\
\Leftrightarrow (\forall a, b : SRule \mid a \in p.Rules \\
\quad \wedge\ b \in p.Rules \wedge a \neq b \\
\bullet \neg\ a\ Redundant\ b \wedge\ \neg\ a\ Contradictory\ b)
\end{array}
$$

$$
\begin{array}{c}
\textbf{theorem}\ verif\_consistency \\
Consistent\ \ Test\_Policy
\end{array}
$$

The theorem's goal predicate is *Consistent Test_Policy*. When, we obtain the predicate true, after running a list of proof scripts, we prove that the conjecture is a theorem, and *Test_Policy*, indeed, is a consistent policy. A detailed example with regard to the proof of an intra-policy consistency is presented in [6].

After specifying and verifying secure mobile agent systems, the Z specification will automatically be transformed to a UML Model, which will be detailed in the next section. This transformation offers to the designer an easily way to design his secure mobile agent system and generate after that the corresponding code. The details of transformation rules is out of scope in this paper.

4 Modeling Secure Mobile Agent Systems

Recently in [9], we proposed a meta-model, which supports the already formally specified concepts required for modeling secure mobile agent systems. Our MDS4MAS meta-model is composed in two packages. The first one describes the mobility concepts while the second one is interested in the security of mobile agent systems. All these concepts are defined in our meta-model, which represents the vocabulary of a language which is used to specify the model. Du to lack of space, we will schematically present, in the following figure, only the security package of our meta-model.

We proposed also an UML profile as an extension of the UML2.0 deployment diagram for modeling secure mobile agent systems. This profile defines all previous cited concepts in our meta-model as UML elements through definition of stereotypes for each meta-class. In addition, our profile contains a set of OCL constraints to impose some restrictions on the defined stereotypes. Figure 3 presents a part of the proposed UML profile. For example, the stationary and the mobile agent extend the meta-class *component*, while the agent system and the mobile agent system extend respectively the meta-class *Execution environment* and *node*. As an example of security elements, the *SecurityModel*, *SecurityPolicy* and *SecurityRule* are defined as *class* and the *SObject* extends the meta-class *Artifact* and *Interface.*

5 AGLETS Specific Modeling

According to the MDA approach, the next step corresponds to the transformation of application model into platform specific model. So, we need essentially to

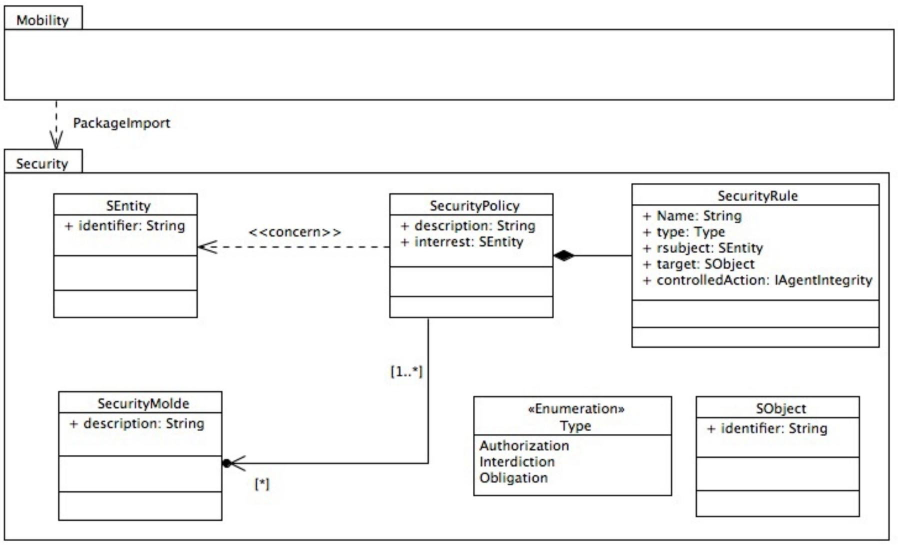


Fig. 2. MDS4MAS meta-model for secure mobile agent systems

specify a deployment platform for mobile agents. We select the AGLETS platform [10] for three main raisons: *First*, AGLETS requires a security framework to ensure the access control of the agents. *Second*, it offers an environment for programming mobile Java objects which react like mobile agents that can move from one machine to another. *Third*, Several concepts of the AGLETS platform are defined in our meta-model.

We proposed an UML profile partially describing the AGLETS platform. It allows to automatically project the application model into AGLETS specific model. This profile is presented in Figure 4, and represents only the most important concepts of the AGLETS platform. These AGLETS concepts are defined as stereotypes. **Aglet** represents the mobile agent, which is identified by an **AgletID** and communicates using **Message**. While **AgletContext** represents the environment of execution which manages the life cycles of the aglets by offering them services and protecting the host against malicious aglets.

We use Atlas Transformation Language (ATL) [11] for automatically translating the application model into AGLETS specific model. We start by defining the mapping between the concepts at meta-models level. The mapping suggested between our MDS4MAS meta-model and the meta-model of AGLETS leads to determine transformation rules needed to be applied on all application models in conformity to the application meta-model.

Fig. 3. The proposed UML profile

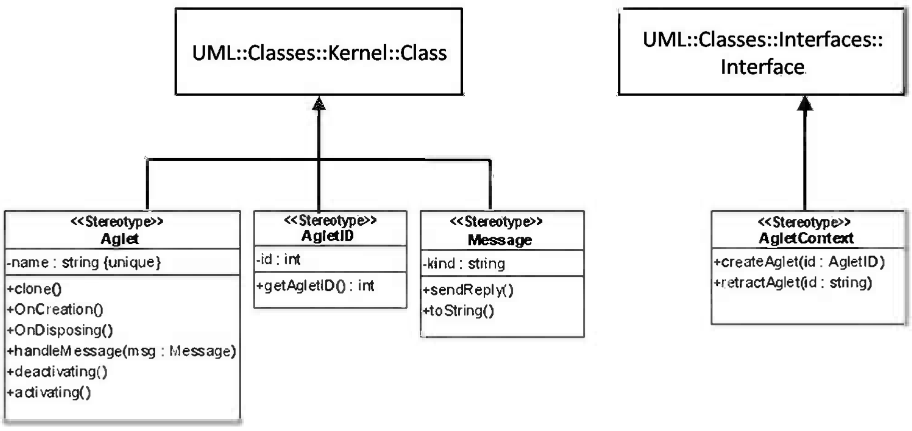


Fig. 4. UML profile for a partial description of AGLETS platform

We take for example the transformation rule named *AgM2Aglet*, which connects the element stereotyped Agent whatever *StationaryAgent* or *MobileAgent* of application model with the element stereotyped *Aglet* of AGLETS specific model. We call a set of lazy rules to add some attributes and operations which are necessary for the implementation of our application, which are AGLETS specific.

6 Code Generation

The code generation process is composed of two parts. The first part corresponds to the generation of the functional code including the mobility concepts. A Java code is automatically generated from the AGLETS specific model, which does not contain any security related code. The second part corresponds to the generation of security code, which is generated automatically from the OCL constraints describing the security constraints. We generate aspect code, which will be integrated in modular way within functional application code to verify at runtime if the specified security constraints are satisfied.

Using Acceleo, we create a new template which takes in consideration all proposed stereotypes in AGLETS specific meta-model. Our template describes that each stereotyped class will be translated as a Java class that inherits another class named by the name of the stereotype. As an example, each Aglet agent (the class agent stereotyped with Aglet) will be translated to a class with the same name and will extend the predefined class Aglet. In addition, all cited methods and attributes in AGLETS specific model will be translated to corresponding code.

We take advantage of the AOP paradigm, which provides a high-level of modularity, to define a generative aspect-based approach that generates the corresponding security aspects. Therefore, we propose to generate AspectJ code to implement security concerns in a mobile agent application. Thus, we define Acceleo template to generate an aspect code from these concerns according to the AspectJ language [12]. We generate an AspectJ aspect for each security constraint specified using OCL.

As presented in Listing 1.1, the aspect template consists of three main parts: an aspect declaration, a pointcut and an advice. (i) The *aspect declaration* (lines 2 – 5) consists of defining the name of the aspect as well as the name of aspect file, which corresponds to the name of the corresponding security constraint. (ii) The *pointcut* (lines 8 – 12) intercepts, for each instance of security rule, the methods execution of the entity concerned with security. These methods represent in our modeling the *controlledAction* attribute in the SecurityRule class. (iii) The *advice* code (lines 15 – 28) checks if the corresponding OCL constraint is specified. If the constraint is well satisfied, the aspect executes the intercepted agent action and updates the state of system. Otherwise, an exception will be handled and the aspect prohibits the execution of this action.

Listing 1.1. Aspect Template

```
// ============== Aspect Declaration (part 1) ==============
[module generateAspect/]
[template public generateAspect(c : Class)]
    [file (c.name.concat('.aj'), false)]
    public aspect [c.name.toUpperFirst()/]{

// ============== Pointcut (part 2) ==============
    pointcut [c.name.concat('pc')/]() : execution (public *
    [for (p : Property | c.getAllAttributes())]
    [if (p.type.name = 'SEntity')] [p.name/] [/if] [/for].
    [for (p : Property | c.getAllAttributes())]
    [if (p.type.name = 'IAgentAction')] [p.name/] [/if] [/for](..));

// ============== Around Advice (part 3) ==============
    around () : [c.name.concat('pc')/]() {
     // Generating Java code from OCL constraints as Java conditions code.
         ....
     // Verifying if all generated constraints are satisfied
            if (allOCLConstraintsAreSatisfieted) {
     // The action will be executed  and the system state will be updated
            proceed();
            ....
            }
     // Otherwise, An exception will be handled
            else{
               system.out.println ("You can not execute this action ...");
            }
     } }
    [/file] [/template]
```

7 Case Study

We implement a graphical editor as an Eclipse plug-in, so that the designer can easily model his secure mobile agent system based on our meta-model. Figure 5 presents the model of our case study E-commerce system, which is composed of a set of stationary and mobile agents. A mobile agent *Buyer* is created in a system agents *SellerSystem1*. This agent can move to another agent systems *to buy* computer resource *Printers*. This agent can move to *SellerSystem2* where exists the *Seller* presented as a stationary agent, responsible for the action to sell the *Printer* resource.

In the shutter console of our MDS4MAS editor, the designer can choose the tab Interactive OCL to add his security constraints by using the OCL language. As an example, we defined the security rule SR1, which prohibits the mobile agent *Buyer* to buy the computer resource Printer with the stationary agent Seller if this Printer is reserved to be sold (Figure 6).

According to our approach, we applied the defined transformation rules to translate the previous platform-independent model (Figure 5) to the corresponding AGLETS specific model, but without supporting security concerns. This model, shown in Figure 7, is more detailed and near to the technical solution, since it will be automatically translated after that to the AGLETS code.

As already explained, the transformation rules automatically generate further methods and attributes for the mobile agent class. For example, in the *Buyer* class, only the method *toBuy* is translated from the application model, all others

Fig. 5. A model of the E-commerce system

methods (e.g., getHome, onCreation, etc.) are automatically created to satisfy the AGLETS requirements for developing mobile agent.

Finally, we apply also the Java and Aspect templates to generate respectively AGLETS functional application code and the corresponding security code as AspectJ aspects. As an example, we present in the Listing 1.2 the *Buyer* class. This code contains first the declaration of the required package as well as the name of the generated class. It contains also the list of attributes (lines 3–5) and the declaration of the methods(lines 6–9).

Listing 1.2. Part of the generated code

```
1  import com.ibm.aglet.*;
2  public class Buyer extends Aglet {
3      private String home = null ;
4      private String message = null ;
5      ...
6      public void toBuy() {...}    // Should be implemented
7      public String getHome() { return this.home;}
8      public void onCreation(Object ini) {...} // Should be implemented
9      ...
10 }
```

Fig. 6. An example of OCL constraint

In the same way, we automatically generate aspect from the OCL constraints, which are defined by the user in the application model. In the following, we detail the generation of AspectJ code from the constraint *SR1* (defined in Figure 5).

In the Listing 1.3, the pointcut *SR1pc* (line 2) intercepts the execution of the public method *toBuy* of the class *Buyer* (the name of this class is defined as the context of the SR1 constraint). We note that the skeleton of this method is already generated as a part of the generation of the functional application code. The advice (lines 3 – 15) associated with the previous pointcut has the type around. The constraint *SR1* is automatically translated to Java condition code (lines 4 – 7). If this constraint is satisfied according to the system state, the method *toBuy* will be executed using the keyword *proceed*(line 8). Otherwise, an exception will be raised and a message will be sent to the user (lines 11 – 14).

Listing 1.3. An example of the generated security aspect

```
1  public aspect SR1{
2    pointcut SR1pc() : execution (public * Buyer.toBuy(..));
3    around () : SR1pc() {
4           if (aClass.target.equals("Printer") && aClass.entity.equals("Buyer")
5          && aClass.controlledAction.equals("toBuy") && aClass.subject.equals("Seller")
6          && ! (aClass.etatres.etat.equals("reserved"))) {
7               ifExpResult1 = aClass.type.equals("Autorization");
8               proceed();
9               // Updating system state.
10
11          } else {
12              ifExpResult1 = aClass.type.equals("Prohibition");
13              system.out.println ("You can not execute this action ...");
14          }
15   }
16 }
```

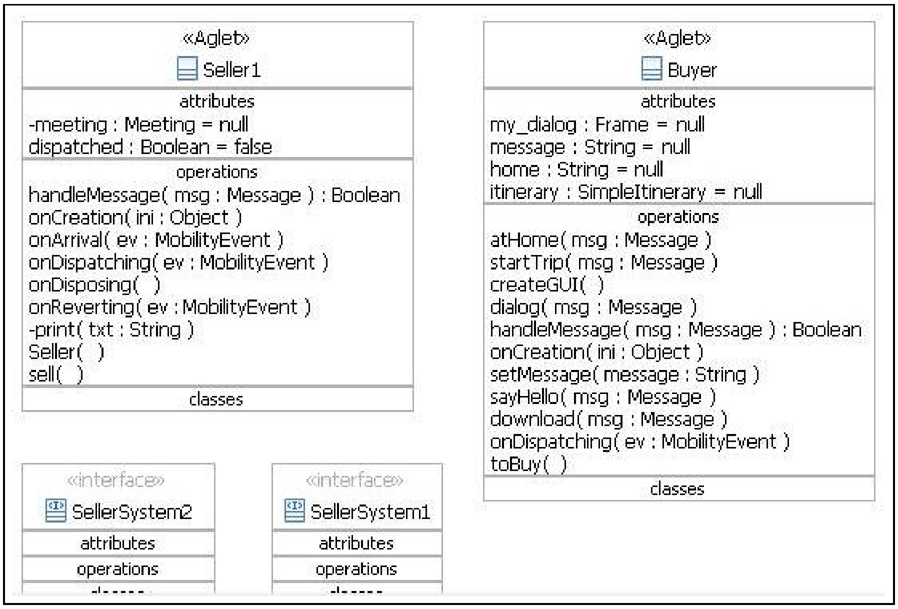

«Aglet»
Seller1
attributes
-meeting : Meeting = null
dispatched : Boolean = false
operations
handleMessage(msg : Message) : Boolean
onCreation(ini : Object)
onArrival(ev : MobilityEvent)
onDispatching(ev : MobilityEvent)
onDisposing( )
onReverting(ev : MobilityEvent)
-print(txt : String)
Seller( )
sell( )
classes
«Aglet»
Buyer
attributes
my_dialog : Frame = null
message : String = null
home : String = null
itinerary : SimpleItinerary = null
operations
atHome(msg : Message)
startTrip(msg : Message)
createGUI( )
dialog(msg : Message)
handleMessage(msg : Message) : Boolean
onCreation(ini : Object)
setMessage(message : String)
sayHello(msg : Message)
download(msg : Message)
onDispatching(ev : MobilityEvent)
toBuy( )
classes
«interface»
SellerSystem2
attributes
operations
«interface»
SellerSystem1
attributes
operations

Fig. 7. Platform-specific model of AGLETS

8 Related Work

Several research works have been proposed for modeling and enforcing security policies for mobile agent systems.

Bryce [2] proposes a security framework which supports the specification of the security policies for mobile agents and their execution systems (host or place). The framework specification has been preceded by the definition of a conceptual framework that exhibits the main concepts related to the system structure, and the different stages of agent life cycle (i.e. creation, communication, migration and termination). Based on these concepts, the security policy of an agent (mobile agent / host) has been expressed. It's limited to the control the agent access rights.

In fact, the rights are assembled into groups and the agent must be associated to a group to benefit from his rights. The agent may have different trust levels at each host. Therefore, it must be adaptable to the execution environment and respond to the trust level of the visited host. This adaptation is expressed by the change of agent group. After a change of group, the agent must ensure that it is able to continue its execution on the new host. To check this property it is necessary to apply logical foundations in the specification of the policy. The use of such foundations was completely absent in this work.

Moreover, this framework is implemented over a Java-based agent system called JavaSeal and it can easily be adapted to other systems. However, the authors do not clearly demonstrate the mapping from models to code. Therefore, it is extremely difficult to use the models created by using the methodologies to generate code to another platform.

Beydoun et al. [13] extend the FAML (FAME Agent-oriented Modeling Language) [14] to support modeling of security concerns. This work constitutes only the first step of the model-driven engineering lifecycle. The proposed extension of FAML consists of two sets of modeling classes (metaclasses): One set to model the security requirements of a multi-agent system (MAS) and another to model security actions satisfying the security requirements. On the one hand, the proposed metamodel does not support the modeling of fundamental concepts of mobile agent systems. On the other hand, the security actions have been modeled in very abstract way. In fact the authors do not explicitly present the possible kinds of actions to undertake, their applicability context, and on which object it will be applied.

Ugurlu and al. [1] describe how to protect system-level resources and agents against unauthorized access. They offer a high level of flexibility for specifying security policies. In this work, two types of policies were supported: The host policy protects local resources from a host against the unauthorized actions. The agent policy determines the aptitude of agent to carry out requests on distant hosts. In addition, this type of policy protects agent against malicious hosts or other malicious agents. Each policy is defined by a number of ECA rules. However, this approach supports only the specification of the access control rules. The platform SECMAP (Secure Mobile Agent Platform), implements this framework. It is a graphical interface that allows the designer to control its agents and manually adapt their policies to the security requirements of the new system. However, the security policies are considered only at the implementation level, which is not completely automated.

In [15], the authors introduce extensions to the Tropos methodology to enable it to model security concerns throughout the whole development process. The authors insert security concepts during the analysis and the design phases. They identify four main modelling activities: the first one allows the flexibility during the development stages of a multi agent system, while the second represents a set of restrictions that do not permit specific actions to prevent certain objectives from being achieved. The third proposed activity involves the analysis of secure goals, tasks and resources identified in a multi agent system, and finally the fourth activity guarantees the satisfaction of the security constraints. Compared to our approach, this work do not clearly demonstrate the mapping from design models to platform specific models and to functional application and security code. Therefore, this work combines formal and informal specifications of MAS. In addition, this work cannot be easily applied into different multi-agent platforms.

Nusrat et al. [16] present a security communication model *SAGLET* basing on existing Aglets architecture. They involve protecting the state of the aglets and their malicious activities. They propose a new service agent along with a specific policy. This service agent allows the authentication of the visiting agents, the control of the communication between service agent and visitor agent, and allocates resource to agent according to the defined policy. This work focuses only on the modeling of the security of the communication between agents and does not provide a framework for specifying the security of the agents and the agent systems as defined in ours. In addition, this work is specific to Aglets and cannot be implemented on several platforms.

9 Conclusion and Future Work

We presented a model-driven approach for implementing secure mobile agent systems. We proposed a framework for modeling mobile agent systems and their security policies. All concepts are formally defined using Z notation. A formal verification is also performed to check the consistency of the model. This platform-independent model is automatically translated to another model specific to AGLETS platform. We proposed also to generate the functional application code from the AGLETS model and generate AspectJ aspects to verify at runtime the specified security constraints. The generated aspects will be integrated in modular way within the functional application code.

Our current work is grounded in applying our approach on other mobile agent platforms using different security aspect languages. The designer should model and formally verify the specification of the secure mobile agent systems independently on the platform. He should also define all security constraints using OCL language.

Based on our approach, we offer to the designer the possibility to generate different platform specific models like Jade, Aglet, Voyager, and Grasshopper and we can also generate the corresponding functional code based on a set of templates. For the security code, we offer the possibility to generate aspect code in different aspect languages, like AspectJ, JAC and JBossAOP.

In this way, the designer can select the mobile agent platform to generate the corresponding code and he can also select the way of weaving the generated security aspects (runtime, compile time, etc), which depends on the selected aspect language.

References

1. Ugurlu, S., Erdogan, N.: A Flexible Policy Architecture for Mobile Agents. In: Wiedermann, J., Tel, G., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2006. LNCS, vol. 3831, pp. 538–547. Springer, Heidelberg (2006)
2. Bryce, C.B.: A Security Framework for a Mobile Agent System. In: Cuppens, F., Deswarte, Y., Gollmann, D., Waidner, M. (eds.) ESORICS 2000. LNCS, vol. 1895, pp. 273–290. Springer, Heidelberg (2000)
3. Maria, B.A.D., da Silva, V.T., de Lucena, C.J.P.: Developing Multi-Agent Systems Based on MDA. In: Proceedings of the 17th Conference on Advanced Information Systems Engineering - CAiSE. CEUR Workshop Proceedings, vol. 161. CEUR-WS.org (2005)
4. Loulou, M., Jmaiel, M., Mosbah, M.: Dynamic Security Framework for Mobile Agent Systems: Specification, Verification and Enforcement. International Journal of Information and Computer Security - IJICS, 321–336 (2009)

5. Acceleo: Effective MDA (2007), `http://www.acceleo.org/`
6. Loulou, M., Kacem, A.H., Jmaiel, M., Mosbah, M.: A Formal Security Framework for Mobile Agent Systems: Specification and Verification. In: Proceedings of the 3rd International Conference on Risks and Security of Internet and Systems, pp. 69–76. IEEE (2008)
7. Woodcock, J., Davies, J.: Using Z: Specification Refinement and Proof. International Thomson Computer Press (1996)
8. Meisels, I., Saaltink, M.: The Z/EVES Reference Manual (for Version 1.5). Technical report, ORA Canada (1997)
9. Rekik, M., Kallel, S., Loulou, M., Kacem, A.H.: Modeling Secure Mobile Agent Systems. In: Jezic, G., Kusek, M., Nguyen, N.-T., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2012. LNCS, vol. 7327, pp. 330–339. Springer, Heidelberg (2012)
10. IBM: Aglets (1996), `http://www.trl.ibm.com/aglets/`
11. Jouault, F., Kurtev, I.: Transforming Models with ATL. In: Bruel, J.-M. (ed.) MoDELS 2005 Workshops. LNCS, vol. 3844, pp. 128–138. Springer, Heidelberg (2006)
12. Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W.G.: An Overview of AspectJ. In: Lindskov Knudsen, J. (ed.) ECOOP 2001. LNCS, vol. 2072, pp. 327–353. Springer, Heidelberg (2001)
13. Beydoun, G., Low, G., Mouratidis, H., Henderson-Sellers, B.: A security-aware metamodel for multi-agent systems (MAS). Information and Software Technology 51, 832–845 (2009)
14. Beydoun, G., Gonzalez-Perez, C., Henderson-Sellers, B., Low, G.: Developing and Evaluating a Generic Metamodel for MAS Work Products. In: Garcia, A., Choren, R., Lucena, C., Giorgini, P., Holvoet, T., Romanovsky, A. (eds.) SELMAS 2005. LNCS, vol. 3914, pp. 126–142. Springer, Heidelberg (2006)
15. Mouratidis, H., Giorgini, P.: Secure Tropos: a Security-Oriented Extension of the Tropos Methodology. International Journal of Software Engineering and Knowledge Engineering 17, 285–309 (2007)
16. Nusrat, E., Ahmed, A.S., Rahman, G.M., Jamal, L.: SAGLET- Secure Agent Communication Model. In: Proceedings of 11th International Conference on Computer and Information Technology - ICCIT, pp. 371–375. IEEE (2008)

Developing Pervasive Agent-Based Applications: A Comparison of Two Coordination Approaches

Inmaculada Ayala[1], Mercedes Amor[1], Lidia Fuentes[1], Marco Mamei[2], and Franco Zambonelli[2]

[1] Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga, Spain
{ayala,pinilla,lff}@lcc.uma.es
[2] Dipartamento di Scienze e Metodi dell'Ingegneria
Università degli studi di Modena e Reggio Emilia, Italy
{marco.mamei,franco.zambonelli}@unimore.it

Abstract. Pervasive computing is concerned with making our lives easier through digital environments that are sensitive, adaptive, and responsive to human needs. Different approaches have shown the suitability of the agent paradigm for the development of pervasive applications. However, so far no dominant approach has been adopted for the development of agent-based pervasive systems. In particular, two key classes of approaches exist, based on FIPA interaction protocols and tuple spaces. The contribution of this paper is the comparison and evaluation of tuple spaces and FIPA-compliant coordination mechanisms for the development of pervasive applications. We are therefore going to compare two approaches that exemplify these agent technologies: MalacaTiny-Sol and SAPERE.

Keywords: Pervasive computing, Agent Platforms, Tuple spaces, Evaluation, FIPA, Aspect Orientation.

1 Introduction

Pervasive computing is about making our lives easier through digital environments that are sensitive, adaptive, and responsive to human needs [1]. Pervasive computing proposes the development of a new generation of advanced systems, in which cheap, interconnected computing devices are ubiquitous and capable of helping users in a range of tasks [2]. Different technologies are contributing to the development of this vision such as distributed computing, mobile computing, human-computer interaction, expert systems or agent technology, just to mention a few.

Different approaches have demonstrated the suitability of the agent paradigm for the development of pervasive applications, because of their capacity to be autonomous, reactive, proactive and social [3,4]. In recent years, pervasive applications based on agents have become a reality, with different projects that exploit agent properties to implement adaptive applications. In these projects, agent technologies have been adapted to these new environments composed of heterogeneous devices and communication means. Agents have been embedded in new devices such as smartphones or sensors, and agent middlewares have been extended to support the heterogeneity, adding new network wireless technologies, and new communication paradigms to facilitate the development of these applications. Agents have been used as abstractions to model and implement both functionality and devices of an Ambient Intelligence systems, to encapsulate artificial intelligence techniques, and to coordinate the different elements that compose the application.

The works presented in [3,4] highlight that a common approach adopted for the development of pervasive systems based on agents still does not exist. In particular, one can think of two radically different models to coordinate agents that compose a pervasive application: indirect coordination based on tuple spaces and coordination based on FIPA interaction protocols. These two options have a great impact on the design of the agent and consequently, on the design of pervasive applications. In tuple-based approaches, agents interact by exchanging tuples, which are ordered collections of information items. Agents communicate, synchronize and cooperate through tuple spaces by storing, reading, and consuming tuples in an associative way [5]. In these approaches, agents have a simpler design because the tuple space embeds most of the logics of coordination. Contrarily, and in accordance with FIPA [6], agents are fundamental actors of a domain, which are able to provide a number of services and provide the functionality of the application and integrate in their code the coordination strategies too. In FIPA approaches, agent communication is based on message passing, where agents communicate by sending individual messages to each other, which are distributed through the Agent Platform (AP).

The contribution of this paper is a detailed comparison and evaluation of tuple spaces and FIPA-compliant approaches for the development of pervasive applications. The goal of this comparison is to illustrate the advantages and disadvantages of these approaches in the development of these applications, and where one approach is more advantageous over another. In order to do this, we are going to use two agent systems that exemplify these agent communication and coordination models: MalacaTiny-Sol [7] as an example of FIPA-based communication, and SAPERE [8] as an example of tuple-based coordination. Using both approaches, we are going to design a few case studies in an Intelligent Museum (IM) in order to compare the resulting systems. In this evaluation we are going to asses the internal design of the agent system, and in addition other important properties of pervasive systems such as adaptability, robustness or privacy. While this paper is grounded on the comparison between MalacaTiny-Sol and SAPERE, we believe most of our analysis can be extended to other FIPA based versus tuple space based implementations.

This paper is structured as follows: Section 2 presents the two approaches that are going to be used in the evaluation, MalacaTiny-Sol and SAPERE. Section 3 describes how to use both approaches to model different scenarios in an IM. Section 4 accomplishes the comparison and evaluation of both approaches and the paper finishes with a Conclusions section.

2 Background

In this section we present MalacaTiny-Sol and SAPERE. These agent systems exemplify two radically different models to coordinate agents that compose a pervasive application: indirect interaction using tuple spaces and direct interaction using interaction protocols. The first one is a traditional coordination model which allows agents to interact uncoupling communicating agents in both time and space by allowing agents to communicate without knowing each other's identities [5]. In order to communicate in an asynchronous way agents read, consume, write or create new tuples in the shared tuple space. The rules (or laws) that govern coordination in the tuple space are defined outside the agents involved.

A large portion of the community considers interaction protocols, i.e. predetermined patterns of interactions, as a means to coordinate MAS [9]. It is the coordination model proposed by FIPA to be supported by FIPA-compliant APs. Internally, as part of their behavior, interacting agents ensure that the message exchange complies with the protocol rules. In order to support the social ability of interacting agents, exchanged messages include the intention of the agent (by means of the so-called performative). Unlike tuple-based coordination, the initiator agent needs to know the identity of its counterpart in the interaction. To support this feature, the AP provides an agent directory facilitator.

Table 1 summarises the main features of MalacaTiny-Sol and SAPERE, showing that they also differentiate in the distribution infrastructure. However, they have some features in common such as the agent architecture type, and some of the devices and network technologies they support.

Table 1. Overview of MalacaTiny-Sol and SAPERE

Feature	MalacaTiny-Sol	Sapere
Agent architecture	Reactive	Reactive
Model of coordination	**FIPA**	**Tuple space**
Supported devices	J2SE-enabled, Android-enabled, **J2ME-enabled**, **Sun SPOT**, **Waspmote**	J2SE-enabled, Android-enabled
Network technologies	**802.11**, 802.15.1, **802.15.4**	802.15.1
Distribution Infrastructure	**Centralized**	**Distributed**

Although the agents of both systems have reactive architectures, their internal design is quite different. The design principle of MalacaTiny is the enhancement of the internal agent architecture, by means of separating the domain specific functionality from other concerns, mainly related with the coordination and exchange of messages. MalacaTiny agents interact according to the FIPA specifications and standards. In SAPERE, agents have a simpler design because the tuple space embeds most the of logics of coordination.

Regarding the requirements of pervasive systems, these approaches focus on different issues. Principally, MalacaTiny-Sol deals with the heterogeneity of devices and communication technologies presented in many pervasive computing scenarios. While SAPERE provides a natural metaphor to develop applications that are distributed in a physical space. More details on these technologies are provided in the following subsections.

2.1 MalacaTiny and Sol

MalacaTiny-Sol is a FIPA compliant agent system, which adapts and extends standard agent technologies to facilitate the development of pervasive applications. In this system we can distinguish two parts: MalacaTiny [10], that allows agents to be developed for lightweight devices; and Sol [7], which is the middleware where these agents are deployed and provides a set of (FIPA) services for those agents (i.e. the AP).

MalacaTiny is an implementation of the Malaca agent architecture [11] for lightweight devices. This agent technology is based on component and aspects[1], which promote the separation of application specific functionality from communication related concerns. In general, in Aspect Oriented (AO) approaches, crosscutting concerns are identified as those concerns that appear to be dispersed in different components of the system, usually tangled with other functionalities. These crosscutting concerns are encapsulated as independent entities named *aspects*. At compilation or runtime, the aspect behavior is again composed at specific points of the system execution described by the so-called *join points* in a process known as *weaving*.

In MalacaTiny these crosscutting concerns are identified in the context of a FIPA-compliant interaction, and are related with the specific functions or tasks that the agent has to perform in order to coordinate with other agents. The considered crosscutting concerns (which are then encapsulated as aspects) are: the formatting of messages (*Representation* aspect), the distribution of the messages using different communication means (*Distribution* aspect), and the coordination, both internal (*Context-awareness* aspect) and external (*Coordination* aspect) for the agent. The join points where these aspects are invoked are the reception and the sending of a message, and event throwing. Aspects are composed at runtime by an aspect weaver ruled by a set of explicit composition rules defined outside of the aspects involved.

The different versions of MalacaTiny are embedded in Android devices, mobile phones with MIDP profile, desktop computers, Sun SPOTs [12] and Libellium waspmotes [13]. MalacaTiny agents can be executed on top of different APs and using different transport protocols, by simply plugging in the correct distribution aspect. For instance, by using the Jade-Leap plug-in, MalacaTiny agents can communicate with other agents registered in this platform. However, current APs for lightweight devices are not entirely capable of managing both device and transport protocol heterogeneity, and have strong limitations to ensure communication interoperability in pervasive systems. The Sol AP has been created to cope with these limitations.

[1] Aspect-Oriented Software Development `http://aosd.net/`

FIPA-based agents require a set of services from the FIPA AP that are related with the transportation of messages between agents, and with the discovering of agents and services. Sol is a FIPA-compliant AP specially well suited to develop applications in the Internet of Things. This AP acts as an agent-based middleware that provides a set of services for the agents and behaves as a gateway to support communication heterogeneity. Specifically, the Sol AP supports:

- The registering and discovering of agents (Agent Management Service-AMS).
- The registering and discovering of services (Directory Facilitator-DF).
- The registration and membership of groups (Group Management Service - GMS).
- The message communication service (MTS), which allows the communication between agents registered in the AP, extended to facilitate the group-based communication.

Note that the AMS, DF and MTS are classic services provided by any AP, but the MTS is extended to support group communication in IoT environments, in conjuntion with the GMS.

Therefore, the main features of this AP (see Fig. 1) are the support for communication of agents in heterogeneous devices, coping with heterogeneous transport protocols (WiFi, Bluetooth and ZigBee) and group communication often required by pervasive systems. Additionally, Sol has remote nodes (Sol Clients in Fig. 1), which communicate with the node in which Sol is running. The development of these clients has been necessary for the implementation of applications distributed in wide areas. Sol clients support devices with low-range communication technology such as mobile phones that use Bluetooth, Sun SPOTs and Libellium waspmotes. These clients can run in desktop computers and Meshlium Xtreme routers [14].

A group is a way to identify a set of agents that are interested in the same type of information. Forming groups enables the Sol AP to implement multicast communication efficiently, which facilitates the distribution of the same information to clustered components of the system. Groups are defined attending to the communication needs of the applications, and agents join and leave these groups at runtime (by the GMS). Groups are usually composed of agents that share some feature (e.g. they are embedded in the same type of device) or play the same role in the MAS (e.g. agents that provide the same service).

In summary, MalacaTiny and Sol (MalacaTiny-Sol) combine to form a system to deal with the requirements imposed by pervasive computing systems. MalacaTiny agents can take advantage of using the Sol AP, so that they can communicate through different transport protocols and send multicast messages to a group of related agents. With this approach, the functionality of the pervasive system is decomposed in a set of MalacaTiny cooperating agents that use the Sol AP for the location and communication between agents. Sol enables interaction via a centralized registration and discovery services. Agents communicate via message exchange and concerns are separated by aspect-based programming via the MalacaTiny framework.

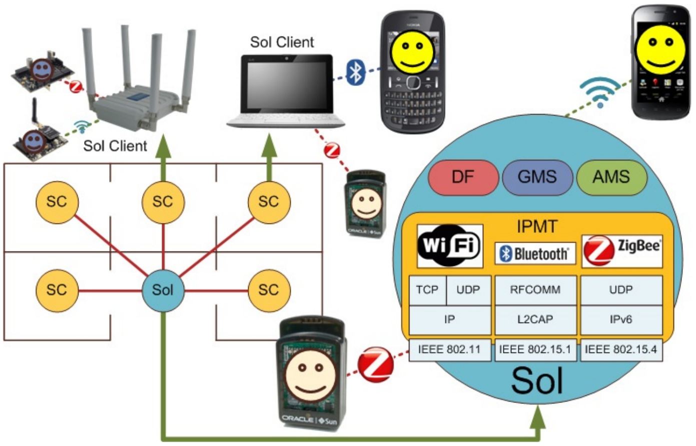


Fig. 1. Schema of the communication in Sol agent platform

2.2 The SAPERE Middleware

SAPERE follows a rather different approach for the development of Multi-Agent applications. SAPERE models a pervasive service environment as a non-layered *spatial substrate*, laid above the actual pervasive network infrastructure. The substrate embeds the basic laws of nature (or *eco-laws*) that rule the activities of the system. It represents the ground on which the components of the pervasive service ecosystem interact and combine with each other. All "entities" living in the ecosystem will have an associated semantic representation: *Live Semantic Annotations* (LSAs), which is a basic ingredient for enabling dynamic unsupervised interactions between components. From an implementation point of view, SAPERE relies on lightweight and minimal middleware infrastructure (see Fig. 2). In particular, it reifies LSAs in the form of tuples, dynamically stored and updated in a system of highly-distributed tuple spaces spread over the nodes of the network [15]. Each LSA acts as an observable interface of resources and service of the components. LSAs of different components can bind with each other to enable interactions. The *eco-laws* are the rules driving the dynamics of the ecosystem. In particular, eco-laws perform pattern matching operations on the set of LSAs that are in the ecosystem to: *(i)* create bindings among LSAs, thus enabling interactions between components, *(ii)* diffuse LSAs across the spatial substrate, *(iii)* aggregate LSAs together, to compute summaries of the LSA population, *(iv)* delete LSAs that are not useful.

The active components of the ecosystem (whether services, software agents, or data sources) express their existence via LSAs injected in the local tuple space associated with their node. Then, they indirectly interact with each other via the tuple space by observing and accessing their own LSA.

In SAPERE, we enforce a notable separation of concerns between application's computation and interaction. Computation (i.e., the main application

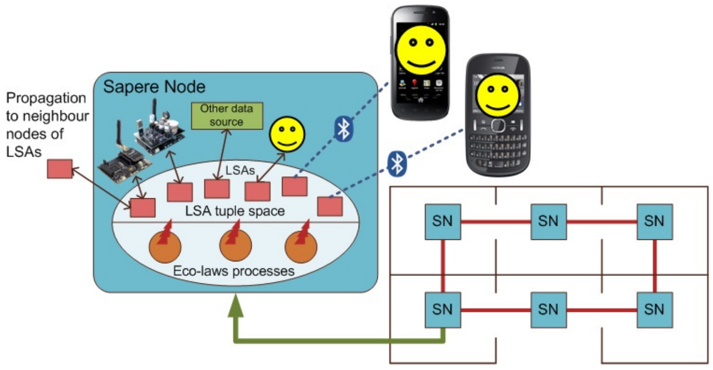


Fig. 2. The SAPERE Conceptual Architecture

business logic) is coded in the SAPERE agents using standard software engineering methodologies. Interaction consists of writing the agents' LSAs and managing their evolution over time. Specifically, programmers have to specify the format of agent's LSAs so that they match with eco-laws, enabling eco-law functionalities: bonding, spreading, aggregating and decaying. In more detail, the eco-laws represent sorts of virtual chemical reactions between LSAs, and are activated by processes embedded in tuple spaces (which make SAPERE tuple spaces different to traditional tuple spaces). Such processes evaluate the potential for establishing new chemical bonds between LSAs, the need for breaking some, or the need for generating new LSAs by combining of existing ones. In addition, to support distributed spatial interactions, eco-laws can enforce the diffusion of LSAs to spatially close tuple spaces, e.g., for those tuple spaces that are neighbor of each other in the network, according to specific propagation patterns (gradient-based diffusion, broadcast, or multicast).

In summary, in SAPERE agents, interactions are mediated by the set of LSA spaces where they inject LSA in the system, and subscribe to the arrival of LSAs. LSAs spread across the network enabling distributed operations.

3 Modeling Pervasive Scenarios

As stated in the introduction, in order to illustrate and evaluate how both approaches work in pervasive systems, we will use an IM, which put together different case study applications. Modern museums' buildings usually include a considerable number of displays and sensors distributed in their rooms, with the goal of providing valuable information to staff and visitors. What characterizes the IM as a pervasive system is the use of sensors and personal devices of people to enhance their experience during the visit. Moreover, the information provided by these devices can be used to improve the efficiency of the running of museum. Specifically, we are going to model scenarios of information provision (Subsection 3.1) and emergency evacuation (Subsection 3.2) in the IM.

Information provision is a very important class of applications used to enrich the IM experience. In particular we focus on: (i) monitoring of the environmental conditions of a room; (ii) controlling the number of people that are currently in the museum; (iii) and the distribution of exhibit information according to a user profile. The first two scenarios are services of interest for the security staff members, while the third is service targeted for museum visitors.

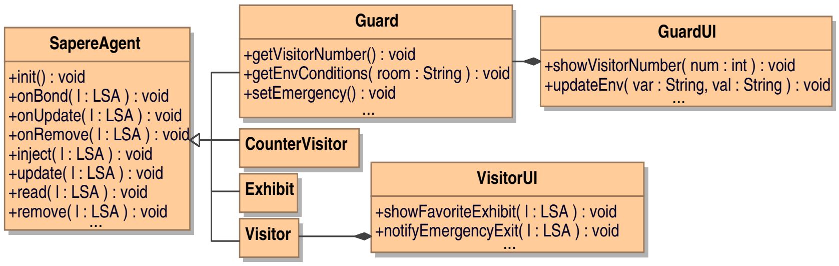
In addition, we are going to model a service that contributes to the evacuation of the building in case of an emergency. This is a a service of great importance in crowded buildings like museums. This problem can be resolved in very different ways according to the characteristics of the two approaches used. In order to illustrate the advantages and disadvantages of both, we are going to consider two situations: there is just one emergency exit; and in the case there is more than one emergency exit.

The design of the above scenarios in MalacaTiny and SAPERE has some points in common. In both systems agents are service providers and consumers and they interact in order to provide services to the people in the museum. Another point in common is in both approaches each guard and visitor have personal agents that are running in their personal devices, and which provide them with the IM services. Additionally, both designs include an agent that represents each exhibit in the museum, and provides information about it. The last point in common is the physical distribution of the middleware because in both solutions they are distributed throughout the building. However, Sol follows the schema depicted in Fig. 1 with a main node and multiple clients, SAPERE follows the schema of Fig. 2 with multiple SAPERE nodes deployed in each room and interconnected.

The main differences are found in the types of agents considered and in the internal design of these agents. In addition to the agents previously mentioned, the MalacaTiny-Sol system incorporates agents to sensors, while the SAPERE system considers a specific agent for counting the number of visitors around a SAPERE node. Although the details of the internal design of agents for the different scenarios will be described in the following subsections, it is important to emphasize that the design of MalacaTiny agents is based on component and aspects, which specify the application functionality and interaction with other agents. So, the description of the architecture of these agents consists of describing the set of components and aspects that compose an agent, and exactly how they relate. However, agents in SAPERE have a very simple design (see Fig. 3) and their behavior emerges from the interactions with the SAPERE node. This interaction depends on the LSAs that the agent injects into the LSAs space and the result of the application of eco-laws to these LSAs. So, the description of these agents is given in terms of injected LSAs and the behavior of the agent when these are bonded, read, removed or updated.

3.1 Information Provision Scenarios

Information provision is a very important class of applications to enrich the IM experience. In particular we focus on: (i) monitoring of the environmental conditions



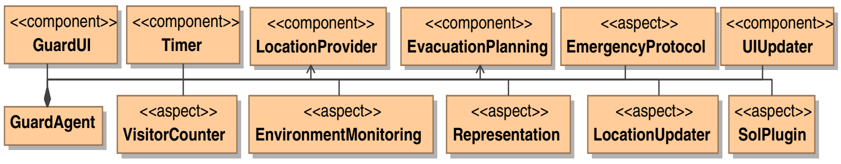
SapereAgent
+init() : void
+onBond(l : LSA) : void
+onUpdate(l : LSA) : void
+onRemove(l : LSA) : void
+inject(l : LSA) : void
+update(l : LSA) : void
+read(l : LSA) : void
+remove(l : LSA) : void
...
Guard
+getVisitorNumber() : void
+getEnvConditions(room : String) : void
+setEmergency() : void
...
GuardUI
+showVisitorNumber(num : int) : void
+updateEnv(var : String, val : String) : void
...
CounterVisitor
Exhibit
Visitor
VisitorUI
+showFavoriteExhibit(l : LSA) : void
+notifyEmergencyExit(l : LSA) : void
...

Fig. 3. UML class diagram of SAPERE agents in the Intelligent Museum

of a room; (ii) controlling the number of people that are currently in the museum; (iii) and the distribution of exhibit information according to a user profile.

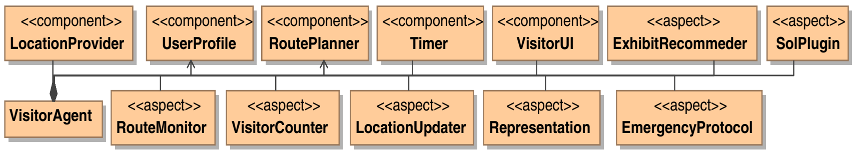
Designing Applications with MalacaTiny-Sol. In these scenarios agents interact to provide information to their corresponding users. In Malacatiny-Sol, agents exchange messages through the Sol AP. This means that the four types of agents that compose the MAS have a distribution and a representation aspect to send and receive messages using the Sol AP named *SolPlugin* and *Representation* (see Figures 4 and 5). To make the interaction between agents more efficient, we define and use groups (introduced in Subsection 2.1). As stated before, with the GMS provided by the AP we can register different groups in order to support the application requirements: one group includes all the visitor agents registered in the AP; another group comprises all the sensor agents that are deployed in a specific room (so there is a group formed for each room with sensors installed); and the last group is for the exhibits that are located in a specific room (so there is a group formed for each exhibition room).

Environmental Monitoring Application. In order to monitor the environmental conditions of rooms, several sensors with agents embedded inside are deployed in them. On initiation, each agent joins the group corresponding to their room. When a security staff member wants to know the conditions in one of the rooms, his agent interacts with the group of sensor agents associated with the room, in order to gather up-to-date information and present the results to the security guard. The implementation of this scenario requires the addition of two aspects and one component (see Fig. 4) to the security guard agent: the *EnvironmentMonitoring* aspect, which requests the information from the sensor group, gathers the answers and updates the internal knowledge of the agent with it; the *UI-Updater* aspect, which updates the user interface with the new environmental results when it observes a change in the agent knowledge; and the *GuardUI* component, which implements the user interface. Components are added to the agent architecture with an identifier using the method *addComponent* (see Fig. 6) and aspects are added by means of aspect composition rules. As stated before, these rules set how aspects are composed at specific points in the agent execution.



<<component>> GuardUI
<<component>> Timer
<<component>> LocationProvider
<<component>> EvacuationPlanning
<<aspect>> EmergencyProtocol
<<component>> UIUpdater
GuardAgent
<<aspect>> VisitorCounter
<<aspect>> EnvironmentMonitoring
<<aspect>> Representation
<<aspect>> LocationUpdater
<<aspect>> SolPlugin

Fig. 4. UML class diagram of the agent for guards



<<component>> LocationProvider
<<component>> UserProfile
<<component>> RoutePlanner
<<component>> Timer
<<component>> VisitorUI
<<aspect>> ExhibitRecommeder
<<aspect>> SolPlugin
VisitorAgent
<<aspect>> RouteMonitor
<<aspect>> VisitorCounter
<<aspect>> LocationUpdater
<<aspect>> Representation
<<aspect>> EmergencyProtocol

Fig. 5. UML class diagram of the agent for visitors

Visitor Counter Application. In order to determine the number of visitors in the IM, the agent for guards must interact with each visitor agent. To make this interaction more efficient, again, we make use of groups. In this case, the guard agent sends an "is alive" request message to the visitor agents group previously defined, and it counts the responses over a time span. To accomplish this task, the security guard agent has to include new components and aspects in its architecture: the *VisitorCounter* coordination aspect, which collects the answers from visitors; and *Timer* component, which determines the time span of the collection. On the other hand, the design of the agent for visitors (see Fig. 5) also includes the aspect *VisitorCounter* that joins the corresponding group at initiation, intercepts the "is alive" request and answers the request of the security agent.

In MalacaTiny, interaction protocol behaviors are implemented as finite state machines whose transitions are driven by internal events or received messages, and that cause the execution of plans. In the case of *VisitorCounter* protocol (see Fig. 7 left side), transitions are driven by events from the user interface that indicate that user requests the number of visitors (*CounterRequestEvent*), messages from visitor agents and the internal event that indicate the end of the time span (*TimerEvent*). Plans of this protocol are: *SendGroupMessage* that sends a message to the group of visitors; *ReceiveAnswer* that processes the answer from visitors and counts the number of visitors (see Fig. 7 right side); and *PresentResults* that presents the results to the security guard.

Information Provision According to User Profile Application. This third scenario provides visitors with information about exhibits in the room where they currently are. The presented information depends on the user personal profile. This scenario requires the visitor agent to know the room where the user is, in order to interact with the agents for exhibits located in the room. The location of the visitor can be obtained internally by the agent using different mechanisms

```
public class GuardAgent extends Agent{
        ....
    protected void setup(){
            ....
        addComponent("UI",guardUI);
        addComponent("Timer",new Timer());
        addComponent("GPS",new LocationProvider());
        addComponent("Evacuation",new EvacuationPlanning());
    }

    protected void compositionRules(){
        addCompositionRule(SND_MSG, Role.REPRESENTATION, ..., AuroraRepresentation.class.getName(),...);
        addCompositionRule(SND_MSG, Role.DISTRIBUTION, ...., SolPlugin.class.getName(), true,....);
                                        ....
        addCompositionRule(RCV_MSG, Role.REPRESENTATION,..., AuroraRepresentation.class.getName(),....);
        addCompositionRule(RCV_MSG, Role.COORDINATION, ...., VisitorCounter.class.getName(),....);
        addCompositionRule(RCV_MSG, Role.COORDINATION, ...., EnvironmentMonitoring.class.getName(),....);
                                        ....
        addCompositionRule(THRW_EVNT,Role.CONTEXT_AWARENESS,...,UIUpdater.class.getName(),....);
        addCompositionRule(THRW_EVNT,Role.CONTEXT_AWARENESS,...,LocationUpdater.class.getName(),....);
                                        ....
    }
}
```

Fig. 6. Partial code of the agent for guards in MalacaTiny

such as the communication network [16]. Each time the visitor moves to another room, the agent changes the group of exhibit agents it has to request the information from. For this purpose, the visitor agent sends a message to this group and when it receives the answers from the exhibit agents, it analyzes the profile of the visitor, and filters the information received to show the information of interest to him/her.

This application is implemented in different aspects of the visitor agent (see Fig. 5): The *LocationProvider* component provides the current location of the agent, notifying a change in the user's position by throwing internal events. The *LocationUpdater* aspect takes the location information, processes it and updates the internal knowledge of the agent with it; and *ExhibitRecommender* aspect ensures that each time the user changes the location to a different room, it interacts with the exhibit agents to gather information and recommend specific exhibits to the user (according to the information in the *UserProfile* component).

Designing Applications with SAPERE. The communication of SAPERE nodes is based on Bluetooth and entities connect to it on a proximity basis. This means that any non mobile element of the IM, like sensors, is automatically connected to the closest SAPERE node. Additionally, in the case of agents embedded in mobile personal devices, they are continuously connecting and disconnecting nodes depending on their proximity to them.

Environmental Monitoring Application. Using SAPERE, sensors accomplish the environmental monitoring and provide it via the injection of LSAs in the SAPERE node that they are connected to. When a guard requests this information, his agent injects LSAs to subscribe to information about environmental conditions. When eco-laws are fired, these LSAs are bonded to the LSAs injected

Visitor Counter Protocol

```
public class VisitorCounter extends CoordinationAspect{
        ...
    protected void setup(){
        ProtocolState initial=new ProtocolState(this,"initial");
        ProtocolState reception=new ProtocolState(this,"reception");

        InstancePattern counterRequest=
                new InstancePattern(new CounterRequestEvent());
        InstancePattern timerEvent=new InsancePattern(new TimerEvent());
        MessagePattern counterProtocolPattern=new MessagePattern();
        groupProtocolPattern.setProtocol("VisitorCounterProtocol");

        registerTransition(counterRequest, initial,
                reception,SendGroupMessage.class.getName());
        registerTransition(counterProtocolPattern, reception, reception,
                ReceiveAnswer.class.getName());
        registerTransition(timerEvent, reception, initial,
                PresentResults.class.getName());

        setInitial_state(initial);
    }
}
```

Receive answer from visitor

```
public class ReceiveAnswer{

    protected void setup(){
        ACLMessage msg=(ACLMessage)getInput();
        Integer visitorCounter=(Integer)getAgent().
                getKnowledge("visitorCounter");
        visitorCounter++;
    }
}
```

Fig. 7. Partial codes of the *VisitorCounter* protocol (left) and the *ReceiveAnswer* plan (right) in MalacaTiny

by sensors and the results are presented to the guard. This application is modeled differently when the security guard is not in the room of which he wants to know the environmental conditions. To do this, it is necessary to have specific agents to gather the conditions and send the information to the remote space when is requested. This procedure is illustrated in the following scenario, when the guard agent wants to know the number of visitors in the IM.

Visitor Counter Application. The modeling of this solution in SAPERE requires the collaboration of three types of agents: security guards, visitors and agents that count the number of people around a SAPERE node. To count all the visitors in the IM it is necessary to know the number of visitors around a SAPERE node and later, to add this information. The interaction between agents for visitors and visitor counter agents is used to determine the number of visitors around a SAPERE node. On the one hand, agents for visitors inject an LSA indicating the presence of their users around the node (see Fig. 8 left) and on the other hand, visitor counter agents are subscribed to this information and update an LSA that contains the current number of visitors around the node (see Fig. 9). These agents increase the counter when LSAs are bonded (user is in the room where the SAPERE node is deployed) and decrease it when they are removed (user leaves the room).

The process for the addition of this information starts with a request of a security guard. Then, his/her agent injects LSAs to request the information injected by visitor counter agents, to do so it has to inject an LSA for each SAPERE node with direct spreading to these nodes (see Fig. 8, left). When these LSAs arrive at their destination, they are updated with the information

Visitor

```
<LSA name="visitor" value="inma"/>
```

Guard

```
<LSA name="museum-visitor" value="0"/>
<LSA name="number-visitor" value="*"/>
<LSA name="number-visitor" value="*" ...
spread="direct" destination="main-hall"
source="room5" .. />
<LSA name="number-visitor" value="*" ...
spread="direct" destination="room3"
source="room5" ... />
                    ...
```

```
Agent Guard {
                                    ...
    int sentLSA,recLSA,visitorCounter;

    onBond(LSA b) {
        if(b.name.equals("number-visitor"){
            visitorCounter=visitorCounter+b.value;
            recLSA++;
            updateLSA(name="museum-visitor",value=counter);
            if(recLSA==sentLSA){
                updateUI("museum-visitor",visitorCounter);
            }
        }
                                    ...
}
```

Fig. 8. Injected LSAs (left) of agents for visitors and guards and partial code *onBond()* method (right) of the agent for guard in the number of visitors scenario

of the number of visitors and sent back to the node of the guard agent. In this node the agent for the guard has injected LSAs to add the values and when it receives all the answers it presents the results to the guard (see Fig. 8, right).

Information Provision According to User Profile Application. The provision of information according to the user profile in SAPERE has an advantage over the solution proposed with MalacaTiny-Sol because it does not have to rely on third components to provide the position of visitors in the IM. When a visitor enters to a new room, his/her agent injects an LSA in the SAPERE node with the personal preferences of the user. On the other hand, agents associated with an exhibit have injected LSAs with information about the exhibit. When eco-laws are fired, the user's LSA is bonded to the exhibit LSAs of interest for him/her and the information is presented to the visitor.

3.2 Scenarios of Emergency Evacuation Planning

In this section, we are going to model a service that contributes to the evacuation of the building in case of an emergency, in both approaches. In order to illustrate the advantages and disadvantages of both, we are going to consider two situations in this scenario: when there is just one emergency exit; and when there is more than one emergency exit available.

Designing Applications with MalacaTiny-Sol

One Emergency Exit Application. The evacuation starts when a member of the security staff detects an emergency situation. Firstly, when an emergency is detected, the security guard agent of the person that detects it, notifies all the people in the IM that there is an emergency situation. To make this notification more efficient, group-based communication is once again used. In this case a message is sent to the group of visitor agents and another to those composed

Visitor counter

```
<LSA name="user" value="*"/>
<LSA name="number-visitor" value="0"/>
...
```

```
Agent VisitorCounter {
                                        ...
  int numberVisitor;

  onBond(LSA b) {
    if(b.name.equals("number-visitor"){
      updateLSA(spreading="direct",destination=b.origin);
    }else{
      numberVisitor++;
      updateLSA(number-visitor=numberVisitor);
    }
  }

  onRemove(LSA b){
    if(b.name.equals("number-visitor"){
      numberVisitor--;
      updateLSA(number-visitor=numberVisitor);
    }
  }
                                        ...
}
```

Fig. 9. Injected LSAs (left side) of the visitor counter agent and partial code of *onBond()* and *onRemove()* methods (right side) of the visitor counter agent

by security agents. With the information provided in the message, visitor agents plan how to get to the emergency exit while avoiding the site of the emergency. Security guards agents use this message to inform the security staff of where the emergency exists and what kind of emergency it is.

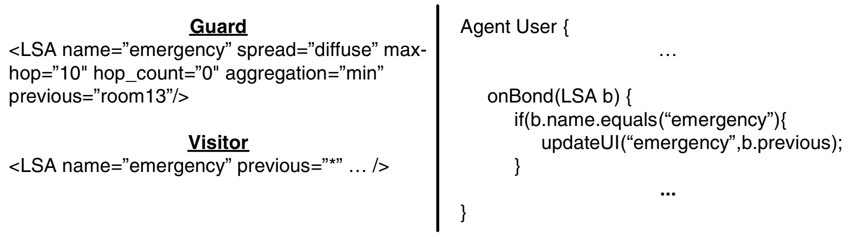
In order to implement this behavior in the security guard agent, new aspects are added (see Fig. 4): the joint work of the *LocationProvider* component and the *LocationUpdater* aspect estimates and updates the user position that is going to be used in the emergency message; and finally, the *EmergencyProtocol* aspect continues with joining the agent to the group of security guard agents, sending an emergency message to the two groups of agents and also receives emergency messages. The design of the visitor agent also requires more elements (see Fig. 5) to manage an emergency situation: the *EmergencyProtocol* aspect receives emergency notifications from security guards and updates the internal knowledge of the agent activating an emergency situation; when this occurs, the *RouteMonitor* aspect requests a route from the *RoutePlanner* component and when the route to the emergency exit is generated, it guides the user to the exit using his/her current location.

Multiple Emergency Exits Application. When there is more than one emergency exit to choose from the situation is similarly handled. As in the previous case, the corresponding security agent sends the message to the group of security agents with the same result. Additionally, it has to determine the number of visitors and their position in the IM. In order to get this information, a message requesting the position of the visitor agents is sent using the group-based communication. When the security agent receives the information it assigns an emergency exit to the visitor according to their current position and the number of visitors in the same room (in order to ensure a speedier evacuation). To implement this behavior the design of the agent for visitors is not modified, but the security guard agent needs to change the behavior of the *EmergencyProtocol* and add a new component for planning the visitors evacuation (*EvacuationPlanning*).

Designing Applications with SAPERE

One Emergency Exit Application. The emergency planning in SAPERE uses the work of the spread and aggregate eco-laws to enable a field-based coordination mechanism [17] that notifies of the emergency and indicates the exit path simultaneously. To trigger this process the agent associated with the guard has to inject an LSA in the SAPERE node located at the emergency exit. This LSA is spread to the other nodes hop-by-hop starting with those that are directly connected to the space in which the LSA was initially injected. When this LSA is spread to other SAPERE nodes, the attribute field is set to the previous node and the hop counter is increased. If multiple emergency LSAs are spread to the same SAPERE node with different origins (i.e. different values of previous attribute), when the aggregate eco-law is fired, only the LSA with the minimum hop-counter remains (see attribute aggregation in Fig. 10). In order for the visitor to receive the emergency notification, his/her associated agent has to inject an LSA in order to receive emergency notifications. The bonding of this LSA enables the planning of the emergency route step by step. Each time the LSA is bonded, the visitor receives a notification of the next room that he/she has to reach to get to the exit of the building. When the user reaches the specified room, the same process is repeated. In this way, visitors find the emergency exit. The application of the aggregation eco-law ensures that visitors always follow the path with the minimum number of hops to the exit.

Multiple Emergency Exits Application. In the case of multiple emergency exits, the security guard agents inject an LSA with the same format as in the previous case and the path with the minimum number of hops is also ensured by aggregation eco-law. The problem is that with this schema we cannot control the number of people that are sent to the different exits. This is because we cannot ensure, on the one hand the minimum path (applying the aggregation eco-law) and on the other hand, to have multiple options that can be used for a specific agent to send a person to one exit or another.



Guard

```
<LSA name="emergency" spread="diffuse" max-
hop="10" hop_count="0" aggregation="min"
previous="room13"/>
```

Visitor

```
<LSA name="emergency" previous="*" ... />
```

```
Agent User {
                    ...

   onBond(LSA b) {
      if(b.name.equals("emergency"){
         updateUI("emergency",b.previous);
      }
                    ...
}
```

Fig. 10. Injected LSAs (left) of agents for visitors and guards and partial code *onBond()* method (right) of the agent for visitors in the emergency evacuation

4 Comparison

In this section we are going to evaluate and compare the systems resulting from the design of the scenarios described in the previous section and modeled using MalacaTiny-Sol and SAPERE. The reason for this comparison is to measure the advantages and the benefits of both approaches from a software engineering point of view. Specifically, this assessment focuses on the design of the security guards and visitor agents. For the evaluation, we are going to use a combination of the frameworks provided by [18,19]. On the one hand, the work in [18] provides an architectural metric suite that is being widely used to measure the separation of concerns in software systems. On the other hand, the paper [19] presents a framework for the evaluation of ubiquitous computing systems, which can be used to evaluate pervasive applications as is our case study. Specifically, we are going to evaluate:

- Separation of Concerns (SoC) - Subsection 4.1
- Coupling and Cohesion - Subsection 4.2
- Adaptivity - Subsection 4.3
- Robustness - Subsection 4.4
- Scalability - Subsection 4.5
- Privacy - Subsection 4.6

In order to implement the metrics for measuring SoC, coupling and cohesion, we are going to use common concerns of MalacaTiny and SAPERE. They are representation, distribution, coordination, context-awareness, bonding, computation and spreading. The mapping between them is depicted in Fig. 11.

4.1 Separation of Concerns

SoC is a well-established principle in software engineering which aims to improve the internal modularity and maintainability of the crosscutting concerns of a software design. A crosscutting concern is a special concern which naturally cuts across the modularity of other concerns. Without the proper means for separation and modularization, crosscutting concerns tend to be scattered

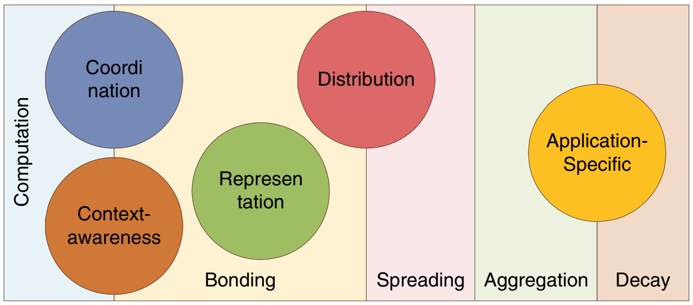


Fig. 11. Mapping between MalacaTiny (circles) and SAPERE concerns (rectangles)

and tangled up with other concerns. The natural consequences are reduced comprehensibility, ease of evolution and reusability of software artifacts, which limit the adaptability, robustness and scalability of the software system.

A way to measure the degree of SoC is to quantify the diffusion of a concern over components, interfaces and operations. Concern Diffusion over Architectural Components (CDAC), Interfaces (CDAI) and Operations (CDAO) measure the degree of concern scattering at different levels of granularity. These metrics count the number of components, interfaces and operations which contribute to the realization of a certain concern and their results are obtained for each concern of interest in the system. The metrics for computing the separation of architectural concerns are applied to calculate the degree to which a single concern or property of the system maps to the architectural components.

The results of the assessment show SAPERE scores better for CDAC and CDAI, while MalacaTiny is better for CDAO (see Table 2). In SAPERE, the agent class encapsulates all concerns except bonding and spreading, that are in the SAPERE node, so their values for CDAC are 0 or 1. In MalacaTiny each concern is encapsulated as an independent aspect, and there is a coordination aspect for each interaction in which the agent participates and context aware behavior (see Figures 4 and 5). The results for the bonding concern for MalacaTiny are because this concern includes coordination, context-awareness, distribution and representation (see Fig. 11). Additionally, the agent interaction is supported in MalacaTiny by 1 interfaces and in SAPERE by 1.

Finally, the interception point model of MalacaTiny and the number of methods that use SAPERE to receive results of the eco-laws application explain CDAO results. MalacaTiny has 3 interception points and each aspect has at least 1 method to access the aspect behavior. For example, the coordination aspect requires 3 operations because it is affected by 2 interception points (i.e. reception and sending of the messages), and has 1 method to access its behavior. On the other hand, SAPERE agents have 4 methods to receive results from the SAPERE node, 4 operations to interact with it and additionally, we have to consider methods in the agent class that call these operations (see Fig. 3). For example, for the computation concern, the agent for guards scores 10 and the agent for visitors 8.

Table 2. Mean of the results of agents for guards and visitors for the SoC metric in MalacaTiny-Sol (M-S) and SAPERE

Concern	CDAC		CDAI		CDAO	
	M-S	SAPERE	M-S	SAPERE	M-S	SAPERE
Context-awareness	2.5	1	2	1	2	9.5
Coordination	2.5	1	2	1	3	9.5
Distribution	1	1	2	1	2	4
Representation	1	1	1	1	2	4
Bonding	7	0	2	0	9	0
Computation	5	1	2	1	5	9.5
Spreading	1	0	2	0	2	0
Average	**2.85**	**0.7**	**1.85**	**0.7**	**3.57**	**5.2**

The results of these metrics describes the main features of the architectures of agents in FIPA-based and tuple-based approaches. Agents in FIPA have a more complex design because the negotiation (coordination in tuple-based approaches) is accomplished inside the agent, so CDAC and CDAI is always higher in these approaches. The results for CDAO are a consequence of the complex interaction that tuple-based agents have with their middlewares. Therefore, despite MalacaTiny-Sol scores are good, considering that it is a FIPA-based approach, SAPERE gets a better SoC.

4.2 Coupling and Cohesion

Coupling and cohesion are two quality attributes of a software design that reflect the quality of a good modularization. Coupling refers to the level of interdependency among the modules (e.g. components) and cohesion is the level of uniformity of concerns of a single module (i.e. the degree of relatedness among the elements -attributes, methods- of a component). A high degree of coupling drastically reduces component reuse, which in turn means poor adaptability. Low cohesion means a concern is spread over different modules, and its evolution as an independent entity will therefore be very difficult to manage. Consequently, it is important to minimize coupling and maximize cohesion in the system design.

The coupling metrics measure the number of components connected to each other. Coupling is evaluated using the Fan in and Fan out metrics for each element of the SAPERE and MalacaTiny agents. These metrics count the number of conventional components which require services from the assessed component (Fan in metric) and the number of components from which the assessed component requires services (Fan out metric).

Table 3. Coupling and Cohesion measurements for MalacaTiny-Sol and SAPERE

MalacaTiny-Sol	Fan in	Fan out	LCC
Application specific components	1	0.55	0
Context-awareness aspect	1	1	3
Coordination aspect	1	1	3
Distribution aspect	1	1	2
Representation aspect	1	1	2
Core agent	9	1	0
Average	2.33	0.925	1.66
SAPERE	Fan in	Fan out	LCC
Application specific components	1	1	0
Core agent	1	1	5
Average	1	1	2.5
Percentage difference	80%	-7.79%	-40%

Table 3 shows the average for coupling and cohesion measurement per component for the two architectures (rows labeled "Average") and the percentage difference of each metric between MalacaTiny-Sol and SAPERE (row labeled as "Percentage Difference"). The positive values of the "Percentage Difference" means lower results for SAPERE, while negative results means lower results for MalacaTiny-Sol. In this case, SAPERE scores better for Fan in because SAPERE agents directly perform less functionality than MalacaTiny agents. On the other hand, MalacaTiny scores better for Fan out because the application specific components of agents have a lower value for this metric.

Cohesion is measured using the Lack of Concern-based Cohesion (LCC). This metric counts the number of different system properties addressed by each class (in SAPERE agents), components and aspects being considered (in MalacaTiny agents). For this metric MalacaTiny scores better (see Table 3). This is because the Malaca architecture focuses on the separation of concerns at the agent level, while SAPERE applies the separation at infrastructure level. So, in the SAPERE agent class all the concerns used in the evaluation are contained with the exception of bonding and spreading that are in the SAPERE node.

With these results we can again see reflected, the architectural features of both types of approaches. MalacaTiny successfully exploits its AO to offer agents that scores better in Fan out and LCC. The scores of MalacaTiny mean a high reusability of the internal components and aspects of agents of this scenario. The good results of SAPERE are supported by the lower number of components required to develop SAPERE agents. Therefore, on the one hand MalacaTiny-Sol efficiently handles complex designs, on the other hand, with SAPERE such designs are not necessary.

4.3 Adaptability

The adaptability metric measures how the system adapts to changes that are external to the application, i.e. changes in user preferences, devices and in the physical space where the IM is located.

The recommendations of exhibits of interest for users depending on their location in the IM (see Subsection 3.1) can be useful for new users and annoying for users that are already familiar with the exhibition. So, a useful functionality that the applications can provide is disabling this service when users request it. To do so, MalacaTiny requires the modification of the aspect composition rules (see Subsection 2.1) so as not to apply *ExhibitRecommender* aspect (see Fig. 5). SAPERE accomplish the same task by not injecting LSAs with user preferences. These procedures can be applied to disable any service that agents provide or consume in both systems.

Sol offers the agents deployed on it the possibility of changing the network interface used for their connection to the AP. As stated before, Sol has support for multiple network interfaces (see Subsection 2.1). An agent can connect to Sol using a WiFi connection, but the agent can change the network interface to Bluetooth in the case of poor coverage. Accomplishing this task does not just require a change in the composition rules. Additionally, the agent has to interact with the AP in order to ensure that it remains in the same groups and provides the same services.

Changes in the museum map have different consequences for the solutions proposed in both approaches. In MalacaTiny, some of the agent services depend on the component *LocationProvider* (see Figures 4 and 5). This component informs which room the user currently is in and depending on its implementation, its substitution could be necessary. SAPERE agents rely on the location to provide services too. The extension of the museum map requires the addition of new SAPERE nodes to these new locations and a change of the routing tables of some SAPERE nodes. The design of the agents remains the same, but the agent for guards has to update its internal knowledge about the IM (see Subsection 3.1).

The modification of the map affects the emergency evacuation (Subsection 3.2) in MalacaTiny. In order to ensure the correct planning of the exit route, it is necessary to modify the *RoutePlanner* component inside the agent for visitors. If the IM has a new emergency exit, it is also necessary to change the *EvacuationPlanning* component. On the other hand, in SAPERE, with the modified infrastructure, the implementation of the emergency planning inside agents remains the same.

Table 4 summarizes the main changes in each of the agent systems to adapt agents and infrastructures. In conclusion, both approaches (FIPA-based and tuple-based) can easily adapt the set of services provided by their agents. MalacaTiny offers more possibilities to adapt the agent architecture. Finally, the adaptation of location-aware services to changes in the physical space requires an extra effort in MalacaTiny, in the case of SAPERE this effort is made at the infrastructure level and requires only small modifications in agents.

Table 4. Issues to change when adaptation is required in MalacaTiny-Sol and SAPERE

Adapted issue	Services provided	Network interface	Deployment space
MalacaTiny-Sol	Composition rules	Composition rules / interaction with Sol	Components and aspects that depend on location
SAPERE	Injected LSAs	Not possible	SAPERE nodes

4.4 Robustness

While adaptability measures how the application deals with external changes, the robustness metric measures how internal events affect the application or the percentage of faults that are invisible to the user. In the two approaches there can be faults both at application level and at infrastructure level. In the case of the application level, if a security guard agent or a visitor agent stops, both MalacaTiny and SAPERE users notice that something is going wrong. However, if the agent that fails is an exhibit one, it only results in the information of the associated exhibit not being presented to the user but the application keeps running. This can also be applied to sensors (see Subsection 3.1).

The robustness of the visitor counter scenario (see Subsection 3.1) is similar in both approaches. In the case of MalacaTiny-Sol, the group mechanism supports the provision of this function. Group communication principally uses IP multicast, which is based on UDP at the application level and is an unreliable transport protocol. This means that message reception is not ensured. This issue can affect the accuracy of the number of visitors obtained in MalacaTiny-Sol. In the case of SAPERE, counting the visitors depends on the visitor counter agents deployed in SAPERE spaces. The accuracy of the information provided by these agents depends on the coverage of the SAPERE spaces and the position of the visitors in the IM. So, as in the case of MalacaTiny, the security guard only receives an estimation of the number of visitors. Additionally, in SAPERE if the visitor counter agent fails, the security guard agent will not receive the information from the room where this agent is deployed. Therefore, the distribution of the functionality between agents and AP in MalacaTiny-Sol provides a more robust application.

At infrastructure level, SAPERE is more robust than Sol. The Sol AP usually runs in a single node with multiple clients that depend on it (see Fig. 1), if the AP fails then those agents cannot interact, register services and join groups and the IM cannot offer services to any of its users. The restarting of the Sol AP affects the entire MAS. However, the distribution of SAPERE nodes means that the failure of a node only affects users that are in the same room (see Fig. 2). Additionally, if the node restarts, only the agents associated with this physical space (agents for sensors, exhibits, to gather environmental conditions and for counting visitors) are affected.

In conclusion, in the IM scenario the distribution of the functionality between agents makes the solution based on FIPA more robust. On the other hand, at infrastructure level both middlewares are special cases. In the case of Sol, their current implementation does not offer a robust infrastructure, but other FIPA-approaches, like Jade, handle the eventual failure of the main node of the middleware. SAPERE is a special case in tuple spaces, given that they usually present a centralized distribution. Therefore, and in order to enhance the robustness of these approaches, the infrastructure of Sol should be modified to make it as robust as other FIPA approaches, in the manner that SAPERE offers a more robust infrastructure, unlike other similar coordination approaches.

4.5 Scalability

The scalability metric measures how the complexity of the system increases when a feature is extended or the system must meet a new requirement. In agent approaches, the extension of a system means the addition of new agents or the modification of an existing one. In this section, we are going to study the scalability of these two agent systems, then studying the effort required to extend the system.

The effort required to add a new agent in the MAS is related with the number of elements that compose the agent and the component reuse. SAPERE agents requires less elements (a mean of 12.5 in MalacaTiny vs. 2 in SAPERE) but the component reuse is higher in MalacaTiny (3 in MalacaTiny vs. 0 in SAPERE).

The extension of an agent to meet new requirements is usually related with the addition of new services to agents other than those which we have initially considered in our design. In this subsection we are going to consider three kinds of services: (i) with a single provider, (ii) with multiple providers; and (iii) global services. Service discovery, invocation and provision is easier in SAPERE because of the application of the bonding eco-law. What in MalacaTiny-Sol is done in three steps (query the DF of the Sol AP, request the service and consume the service) in SAPERE is done in two steps (inject the LSA and receive the service).

In MalacaTiny-Sol, the addition of a service with a single provider requires the addition of new aspects and components, and the modification of the aspect composition rules. In SAPERE, this requires the injection of new LSAs and the modification of the agent class to provide or consume the service. To promote the code reuse in SAPERE, the class of the initial agent is extended. In both approaches this extension just implies the modification of the core agent class, however these modifications are easier in MalacaTiny because it ensures the code reuse. To the contrary, the code reuse is more problematic in SAPERE, and consequently so is the extension of the agent. This is because we cannot extend the agent with services provided by more than one agent because multiple inheritance is forbidden in Java. Moreover, to directly use the code is difficult because this is spread between the different methods of the agent (see Table 2).

The addition of a service with multiple providers is different in the two approaches. In MalacaTiny-Sol, agents need a protocol to select the most adequate provider according to some criteria (e.g. using a FIPA Contract Net) which makes the design of the coordination aspect more complex. On the other hand, the case of SAPERE is simpler because in some cases it is the LSA space which selects the most adequate service provider. If the selection criteria is numerical and the most adequate service provider is that which has the minimum or the maximum value, then the LSA space selects the most adequate service provider by means of the aggregation eco-law. If the criteria is not numerical and only implies the existence of a specific feature, it is necessary to include this feature in the LSAs that publish and request the service. If more than one LSA shares this feature, one of them is randomly selected for bonding.

In the case of global services, i.e. services available in any room of the IM, the design of the MalacaTiny does not require any special consideration, but the the SAPERE solution must be modified. This was illustrated in the case of the visitor counter application, that has to deploy purpose specific agents in each SAPERE node. These agents gather the information related with the service and send it (under request) to the interested agents.

The explanations in this section exemplify the work of both coordination approaches in scalability. Without a proper modularization that promotes the code reuse, the addition of new agents requires less effort in tuple-based approaches because the design of agents is simpler. On the other hand, the extension of agents, which usually implies the addition of new services provided or consumed by them, is usually easier to design in FIPA approaches, while in tuple-based approaches the provision and consumption is easier. In the case of these two agent systems, we can conclude that the solution provided by MalacaTiny-Sol is more scalable. MalacaTiny offers a uniform solution for the extension of agent capabilities and promotes software reuse which decreases the development effort. The strongest point in favor of SAPERE is the development of services based on local interactions, which are very likely to be found in pervasive environments, which are provided and consumed by means of the bonding eco-law.

4.6 Privacy

The privacy metric evaluates the type of information that the user has to provide (and divulge) in order to profit from the application, and the availability of the user's information, for other users of the system as third parties. In the scenarios presented, users share three types of data information: presence, location and personal profile. Table 5 depicts what kind of user information is shared by the security guard agents (G) and the visitor agents (V) in the different scenarios. According to the table, scenarios modeled in MalacaTiny-Sol require less information to be shared (user profile remains inside the personal user agent) than those modeled in SAPERE. Additionally, in the case of MalacaTiny the personal information of the users is located in their personal devices, while in the case of SAPERE this information is located in these devices and in SAPERE spaces, where the information can be accessed (through bonding) by all the agents of the IM.

Therefore, we can conclude that MalacaTiny-Sol scores better in privacy because users divulge less information to obtain value from the application and the availability of the information to other users is lower. These scenarios illustrates the work of FIPA-based and tuple-based approaches in privacy. Local computation of FIPA approaches makes it easier to ensure the privacy of users.

Table 5. Type of information shared by agents for guards (G) and for visitors (V) in scenarios of the IM

Scenario	MalacaTiny-Sol			SAPERE		
	Presence	Location	Personal	Presence	Location	Personal
Environmental monitoring						
Visitor counter	V			V	G	
Exhibit information						V
Single emergency exit		G		G,V		
Multiple emergency exit		G,V		G,V		

5 Conclusion

In this paper we have presented the modeling of a classic pervasive scenario, an IM, using two agent systems for pervasive computing, MalacaTiny-Sol and SAPERE. The first one is based on FIPA interaction protocols and the second one is based on tuple spaces. The resulting systems have been evaluated using an architectural metric suite that measures SoC, Coupling and Cohesion, and additionally, we have discussed other important concerns in pervasive systems such as adaptability, robustness, scalability and privacy.

Results of the architectural suite are specific to this case study, but these highlight the architectural features of both agent technologies and also support the following argument. The benefits from both approaches for the development of pervasive application come not only from their schemas of interaction, but also from their mechanisms for ensure SoC (i.e. AO vs. eco-laws) and how they adapt the agent paradigm to pervasive computing (e.g. groups vs. distributed tuple spaces).

In MalacaTiny-Sol, the design of the agents is more adaptable, scalable and can ensure the privacy of users easily. FIPA-based approaches allow the set of offered services to be modified by enabling or disabling behaviors that perform such services. MalacaTiny-Sol does this efficiently even at runtime because these behaviors are encapsulated in aspects whose functionality is driven by a set of composition rules. Additionally, MalacaTiny agents can utilize the multiple network interfaces offered by Sol because of the AO. This is an advantage for the development of leisure applications where users with different types of network technologies in their devices (e.g. Bluetooth or WiFi) can interact with the same system. In general, FIPA-based approaches offer solutions that are more scalable because the extension of the system is uniform. Additionally, because the computing is encapsulated inside agents the risk of lateral effect when the system is extended is lower than in tuple based approaches such as SAPERE. Finally, in FIPA-based approaches it is easier to ensure the privacy of users, because most of the computation is performed locally.

SAPERE agents have a good capacity for adaptation too, their service consumption and provision is easier and additionally, the resulting system is more robust. While in FIPA-approaches adaption of services is related to the number of protocols that agents can handle, in tuple based approaches the relation is with the number of tuples which are injected in the space. So, the adaptation of both types of approaches presents a similar difficulty. The distributed nature of SAPERE applications results in systems that adapt easily to changes in the physical space where the application is distributed. In tuple based approaches the service provision and consumption is more efficient than in FIPA ones because a direct interaction between agents is unnecessary. The negotiation is done in the tuple space via a pattern matching process that avoids message exchange. Finally, SAPERE spaces offers a more robust infrastructure thanks to its multiple SAPERE nodes. This is not a common feature in tuple based approaches but it is one of the strongest points of SAPERE.

The two approaches can be combined with benefits to both. The AO of MalacaTiny agents and its extensible join point model makes the deployment of these agents in SAPERE nodes possible. In this combination, MalacaTiny becomes a tuple-based agent. To do this, it is necessary to develop a distribution aspect for SAPERE and to add 4 interception points in the agent that correspond with bonding, reading, removing and updating of the LSAs. The main benefits for MalacaTiny is the usage of an infrastructure which is more robust than Sol and offers a natural metaphor to develop services that depend on the location of users. The main benefits for SAPERE would be to enhance the internal modularization of agents deployed in SAPERE nodes, that promote reuse and ease the adaptation of agents even at runtime. It is interesting to note that these benefits are not related with features of the coordination approaches, but rather with adaptations of these agent technologies to the pervasive computing environment. As future work, we plan to study the combination of these approaches.

Acknowledgment. Work supported by the Andalusian regional project FamWare P09-TIC-5231, the European projects INTER-TRUST FP7-317731 and SAPERE FP7-256873, and the Spanish Ministry Projects RAP TIN2008-01942 and MAVI TIN2012-34840.

References

1. Saha, D., Mukherjee, A.: Pervasive computing: a paradigm for the 21st century. Computer 36(3), 25–31 (2003)
2. Henricksen, K., Indulska, J., Rakotonirainy, A.: Modeling context information in pervasive computing systems. In: Mattern, F., Naghshineh, M. (eds.) PERVASIVE 2002. LNCS, vol. 2414, pp. 167–180. Springer, Heidelberg (2002)
3. Sadri, F.: Ambient intelligence: A survey. ACM Comput. Surv. 43(4), 36:1–36:66 (2011)
4. Cook, D.J., Augusto, J.C., Jakkula, V.R.: Ambient intelligence: Technologies, applications, and opportunities. Pervasive and Mobile Computing 5(4), 277–298 (2009)
5. Omicini, A., Denti, E.: From tuple spaces to tuple centres. Science of Computer Programming 41(3), 277–294 (2001)

6. FIPA: The Foundation for Intelligent Physical Agents, `http://www.fipa.org/`
7. Ayala, I., Amor, M., Fuentes, L.: An agent platform for self-configuring agents in the internet of things. In: Third International Workshop on Infrastructures and Tools for Multiagent Systems, ITMAS 2012, pp. 65–78 (2012)
8. Castelli, G., Mamei, M., Rosi, A., Zambonelli, F.: Pervasive middleware goes social: The sapere approach. In: Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW), pp. 9–14 (October 2011)
9. Labrou, Y., Finin, T., Peng, Y.: The current landscape of agent communication languages. Intelligent Systems 14, 45–52 (1999)
10. Ayala, I., Amor, M., Fuentes, L.: Self-configuring agents for ambient assisted living applications. Personal and Ubiquitous Computing, 1–11 (2012)
11. Amor, M., Fuentes, L.: Malaca: A component and aspect-oriented agent architecture. Information and Software Technology 51(6), 1052–1065 (2009)
12. Oracle: Sun SPOT world, `http://www.sunspotworld.com/`
13. Libellium: Waspmote, `http://www.libelium.com/products/waspmote`
14. Libellium: Meshlium Xtreme, `http://www.libelium.com/products/meshlium`
15. Mamei, M., Zambonelli, F.: Programming pervasive and mobile computing applications: the tota approach. ACM Trans. Software Engineering and Methodology 18(4) (2009)
16. Rodriguez, M., Favela, J., Martinez, E., Munoz, M.: Location-aware access to hospital information and services. IEEE Transactions on Information Technology in Biomedicine 8(4), 448–455 (2004)
17. Mamei, M., Zambonelli, F.: Field-Based Coordination for Pervasive Multiagent Systems, 1st edn. Springer Publishing Company Incorporated (2010)
18. Sant'Anna, C., Lobato, C., Kulesza, U., Garcia, A., Chavez, C., Lucena, C.: On the quantitative assessment of modular multi-agent system architectures. NetObjectDays (MASSA) 224 (2006)
19. Scholtz, J., Consolvo, S.: Toward a framework for evaluating ubiquitous computing applications. IEEE Pervasive Computing 3(2), 82–88 (2004)

Agent Perception within CIGA: Performance Optimizations and Analysis

Joost van Oijen[1,2], Han La Poutré[1,3], and Frank Dignum[1]

[1] Utrecht University, Utrecht, The Netherlands
{J.vanOijen,J.A.LaPoutre,F.P.M.Dignum}@uu.nl
[2] VSTEP, Rotterdam, The Netherlands
[3] CWI, Amsterdam, The Netherlands

Abstract. When agents are integrated in a game engine for embodiment in a virtual environment, perception often leads to performance issues due to the lack of control over the sensing process. In previous work a perception framework was proposed within CIGA, a middleware facilitating the coupling between a multiagent system and a game engine. It allowed agents to have control over the flow of sensory information generated in their embodiment. In this paper we continue this work by presenting performance optimizations within this framework. Here, the computational complexity of the sensing process in the game engine can be controlled by an agent itself, allowing it to deal with more complex environments. Additionally we provide an overall performance analysis of the framework.

Keywords: Virtual Agents, Middleware, Perception, Performance.

1 Introduction

In games, simulations or training, virtual environments are becoming increasingly more realistic, complex and dynamic. With this comes the need to populate these environments with virtual humans. The use of BDI-agents seems a good fit to realize intelligent behavior for virtual humans.

Using BDI-agents to control the behavior of virtual humans typically involves coupling a multi-agent system (MAS) to a game engine, where an agent's action-selection and perception mechanisms in the MAS have to be connected to the actuators and sensors of its virtual embodiment in the game engine. Successful connections have been made before [6,15,5], though usually in an ad-hoc manner employing a primitive sense-act interface. In these connections, one has to comply to the fixed abstraction level of the sense-act interface, as determined by the specific game engine that was used. Besides this constraint, perception is typically a one-directional process: the embodiment of the agent in the game engine fully controls what kind of percepts are communicated to the agent and how often. This can lead to two problems: first of all, without any control over the flow of percepts, an agent can become flooded with sensory information, stalling its deliberation. Also, the sensing process in the game engine alone is often a costly operation where continuously is decided what can be sensed (e.g. visual or audible signals depending on the sensor's modality) (e.g. [9]). These aspects easily lead to performance issues when considering more complex environments than typically previously considered. Second, information provided by the game engine as sensory information may not be suitable for an agent to reason efficiently: where game engines process data at the geometrical level, agent deliberation typically takes place at a symbolic level where environment knowledge is represented at a strategic abstraction level from the actual state of the virtual environment, better suited for high-level decision-making.

Tackling the above problems related to agent perception, in previous work [17], a perception framework was proposed within the scope of CIGA[1], a general-purpose middleware designed to facilitate the coupling between MASs and game engines [18]. The framework was built around two main concepts: First, it introduces a semantic world model as a representation of the virtual environment defined at a certain abstraction level. This model can be user-defined for a specific domain. Synchronization with the actual state of the environment (i.e. the game state) is achieved through user-defined semantic translations to translate elements from the game state to concepts from the semantic world model. Second, agents within CIGA are equipped with a subscription-based filtering mechanism that can be used to fully control their flow of sensory information. Here, a subscription represents an agent's cognitive interest towards certain environment information. During an agent's sensing process these subscriptions can be taken into account to control its access to the semantic world model.

Based on evaluation results [17], the framework was able to control the flow of sensory information for an agent using subscription-based filtering, though, when dealing with heavily populated environments, a performance bottleneck was seen, caused by the physical sensor processing in the game engine.

In this paper we explore several performance optimizations for the perception framework within CIGA: We (1) investigate a design approach to limit physical sensing processing within a game engine taking into account an agent's perceptual demands controlled from a MAS; (2) investigate the use of *caching* within the semantic world model to limit the amount of semantic translations to be made for creating fresh sensory information; (3) provide an analysis of relevant processes within the framework to identify a next performance bottleneck.

The paper is structured as follows. First, in Section 2 related work is presented. Following, Section 3 and Section 4 describes the CIGA middleware and outlines its perception framework respectively. Several optimizations to this framework are proposed in Section 5. In Section 6 these optimizations are evaluated while in Section 7 a final analysis is provided. Finally, in Section 8 we conclude on our findings.

2 Related Work

Our work covers several research areas including sensor modeling in virtual characters, active perception in BDI-agents and the connection between multi-agent

[1] Creating Intelligent Games with Agents

systems and game engines. Our work is novel in the sense that we have not seen similar work been done combining all of these areas within a general framework for realizing controlled perception for BDI-agents embodied in a real-time virtual environment. Therefore, below we describe some related work in specific areas, partially related to our work.

2.1 Situated Multiagent Systems

Considering agent perception in MASs, Maes pointed out the poor support for active and goal-driven perception in MASs [10]. This issue has been tackled by several researchers resulting in different approaches towards active perception. Weyns et al. [19] present a general model for active perception for situated multiagent systems in which an agent can direct its perception to relevant aspects in the environment, related to the agents' current task. In the model, three stages are employed: First, the *sensing* stage maps the environment state to representations of symbols, influenced by sensor properties denoted as *foci* (e.g. modality, resolution or range) and domain-dependent constraints denoted as *perceptual laws* (e.g. one cannot perceive objects obscured by other objects). Next, the *interpretation* stage maps these representations to higher-level percepts understood by the agent internals (e.g. treat objects near to each other as a group). Finally, the *filtering* stage filters these percepts based on specific selection criteria (e.g. only pass percepts related to other agents). In our previous work [17], we have proposed a subscription-based filtering mechanism that can be compared to the *filtering* stage to filter percepts that will be processed by the agent. Further, the *sensing* and *interpretation* stage is related to our sensory processing: perceptual laws and foci are realized by an agent's virtual sensors (through sensory queries on the game state based on *physical* sensor properties); and interpretation is based on domain-dependent semantic translations to convert game state representations to ontological percepts.

So and Sonenberg [14] relate their work with Weyns et al. and extend it within the context of situation awareness (SA) according to Endsley [4]. They present situation awareness as a process working between perception on one side and practical reasoning on the other side. Like Weyns et al. they include the notion of top-down perceptual attention and use event calculus for specifying this focus of attention, influencing the perception process.

Van der Vecht [16] addresses perceptual attention as adjustable autonomy which it defines as having a dynamic control over external influences on the decision-making process based on internal motivations. It separates event processing from decision-making and introduces reasoning rules to specify the effects of external events from the environment on the beliefs and goals of the agent.

In all the above work, cognitive filtering of sensory information takes place in the MAS agent, after physical sensing and the creation of percepts as symbolic representations. We state that following such a sequential approach of filtering sensory information can lead to serious performance issues when agents are embodied in a virtual environment in a game engine. In complex virtual environments there can be an enormous amount of information that can be sensed by an agent, risking the violation of real-time performance constraints. We will propose an approach in which an agent's cognitive behavior can influence its physical sensing process in order to deal with the computational complexity of an agent's sensor.

2.2 Virtual Human-Like Sensing

Synthetic sensing refers to computational models or techniques employed to simulate human-like sensory abilities like vision, hearing or touch. Out of the human-like senses, vision is often regarded as the most important to simulate. In [11] it is stated that synthetic vision can range from geometrical approaches to pure synthetic vision where the environment is being rendered from an agent's point of view. Geometrical approaches make use of range or collision tests and ray-casting. For example the use of viewing volumes centered around an agent or the use of one or more viewing cones originating from the eye's position allows one to calculate physical entities within sensory range of an agent (e.g. [12,9]). Ray-casting techniques are used to determine if an entity can really be seen and not blocked by other geometry. More advanced models are described in [8], where sensing is based on key concepts of the Spatial Model of Interaction (SMI), a successful awareness model, by taking human concepts into account. In our work, we do not impose strong rules on the implementation of a sensor. This allows for a range of different sensing techniques to be employed.

To facilitate agents in perceiving their environment several approaches have been proposed that involve introducing a semantic layer on top of the actual virtual environment. For example, the cognitive middle layer used in [3]; the semantic virtual environment introduced in [7]; or the use of state detectors in a middleware in [13]. In these approaches, little is said on the performance aspects of maintaining such a layer with respect to the actual game state. Further, the focus is mainly on the management of the semantic layer and individual agent perception through virtual sensors is not considered.

2.3 Connecting MASs to Game Engines

Finally we consider the work on the integration of multiagent platforms or other decision-making systems in virtual environments. In [5], the cognitive BDI-architecture of CoJACK was used to control characters in VBS2, a 3D training environment used in military domains. Pogamut [6] is designed as a mediation-layer between a game engine and an agent's decision-making system to bridge the "representational gap". In [15], the agent programming language GOAL was used to integrate BDI agents in the UT game engine using both Pogamut and EIS. The latter is a proposal for an environment interface standard for MAS agents and has been advertised for use in agent platforms including 2APL, Jadex or Jason [2].

In the above work, although performance issues have been recognized for agent sensing, no approach is presented on how to tackle this issue. This is due to the fact that the specific game engines that were used (e.g. VBS2, Unreal

Engine) or the external interface they were dependent on (e.g. Gamebots[1]) did not offer any facility to control or influence the percept generation process. In previous work, we have proposed a design approach for interfacing MASs with game engines and proposed a technique to control the percept generation process in a game engine [18,17]. Our current work builds on this design approach by introducing optimizations for agent perception.

3 CIGA Middleware

In this section we describe CIGA, a middleware to facilitate the coupling between a multiagent system (MAS) and a game engine in order to build intelligent virtual agents (IVAs) [18]. Its motivation stems from the idea of employing existing agent technology developed by the MAS community to deal with the decision-making aspects required for virtual characters (e.g. as BDI-agents). We start below with a summary of fundamental design issues one is faced with when connecting MASs to game engines and describe how CIGA facilitates in tackling these. For the scope of this paper we focus in particular on the issues related to agent perception. Afterwards, the high-level architecture of CIGA is discussed.

3.1 Design Issues

The first and probably the most significant issue for interfacing MASs to game engines is to bridge the representational gap between the two system. This gap is caused by the difference in the primary function of each system. When considering game engines, these are specialized in realizing physical simulations of virtual worlds and the human-like entities that populate these worlds. Multiagent systems on the other hand are specialized in realizing cognitive simulations driven by cognitive agents and their interactions. In each system, specialized components are typically available that have become more or less standard elements of either technology. In a game engine this is typically a rendering engine to visualize the current state of the environment; a physical engine to simulate the laws of physics; and an animation engine to allow human-like bodily movements. In a MAS, agent designers often employ specialized techniques and models related to knowledge inference, planning or communication. Now where game engine techniques work with data at the more physical and geometrical level, techniques used in MASs work with data at the more symbolic, social level, based on human-like notions. This difference in data representation becomes an issue when one attempts to interface these systems to realize an embodied cognitive agent that can sense and act within a virtual environment. To give an example, an agent's concept of a 'closed door' could be represented in the game engine by geometry in the shape of a door together with the relative rotation with respect to adjacent geometry. Instead of making an agent responsible for interpreting such low-level information, it may be better suited to present an agent with information at a more strategic abstraction level from the actual data representations in the virtual environment.

The second issue relates to situating a software agent within a real-time virtual environment, embodying a human-like character. MASs are generally not designed to connect to real-time virtual environments, nor do they typically require agents to exhibit natural, human-like behavior in these environments. For example, commonly an agent in a MAS receives information from its environment as so-called percepts. In deliberative software agents like BDI-agents these percepts are generally taken into account during deliberation, updating an agent's belief base or triggering a reactive action. The nature of the environments in which such software agents are situated is often more discrete and event-driven. In comparison, in a game engine, the dynamics of an environment is more continuous, driven by a simulation update cycle (often running at rates of more than 30 Hz). Now if an agent becomes embodied in such a real-time virtual environment there can be a huge amount of information that the game engine can provide to the agent as sensory information (e.g. consider a heavily populated environment where each object may have dozens of properties). Of course, for each agent this information should be filtered according to what it can physically perceive through its virtual sensors, which are presumed to be present in the game engine. Still, the amount of information that can considered as sensory information can be enormous, especially in complex and dynamic virtual environments. Additionally, such sensory information can be provided to an agent at each simulation update cycle. The amount of sensory information and the frequency at which this information can be transferred to an agent can put a high burden on the communication interface between an agent and its embodiment in a game engine, leading to a potential performance bottleneck in applications with many agents. Additionally, perception could lead to performance issues for an agent's deliberation cycle. Because of the high amount of percepts being received continuously, an agent's inference engine or rule-based decision-making engine may need to be consulted many times per second, triggered by a continuous stream of up to date information sensed from the environment. This could stall deliberation where an agent becomes occupied with processing information rather than actually planning actions.

The CIGA middleware is designed to facilitate designers to tackle the above issues in a structured manner. Concrete design approaches concerning these issues are described in Section 4. Additional information on the high-level architecture and technical design of CIGA can be found in [18].

4 Agent Perception Framework

In this section we describe the perception framework which illustrates how CIGA tackles the design issues described in the previous section. Figure 1 illustrates the perception framework. To summarize the role of CIGA: on one hand it offers agents access to a global semantic world model which presents an abstract representation of the actual virtual environment. Such a world model consists of semantic concepts, used by an agent's embodiment to create percepts. On the other hand, it offers agents a way to control percept generation through the use of a subscription-based filtering mechanism. Here, a *subscription* represents an interest in certain environment information, relating to the concepts available in the semantic world model. In this approach, agent designers have full control over the flow of percepts from a game engine to a MAS. Agents can be designed to adopt or retract subscriptions, dependent on their current task or goal. Important to note is that although the semantic world model is global within the system and accessible by agents, agents are not omniscient. Access to the world model is regulated by one or more agent sensors, controlled on one hand by a sensor's physical ability to observe entities in the environment and on the other hand by an agent's cognitive interests in certain environment information. Below we describe the framework in more detail.

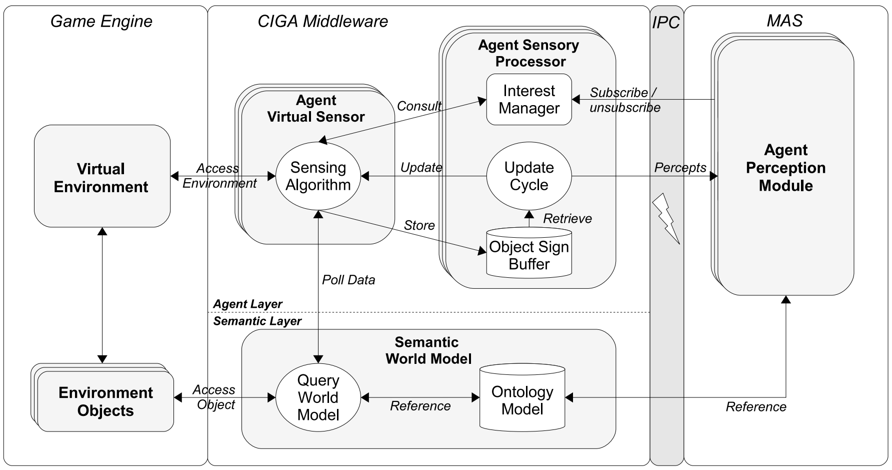


Fig. 1. CIGA Agent Perception Framework

4.1 Semantic Layer

CIGA's *Semantic Layer* in Figure 1 consists of a semantic world model (*SWM*) which agents can access during sensing to obtain sensory information (reflecting their current view of the environment). The *SWM* represents the current world state of the environment at a certain abstraction level from the actual *game state* in the game engine. This abstraction level should be defined at a strategic level for agents to perform efficient decision-making based on meaningful concepts.

The *SWM* consists of instances based on an ontology of environmental concepts that can be specified at design time for a specific domain. Here one can define *object classes with properties* and *parameterized events*. At runtime, this model can be accessed by agents during sensing to request semantic object properties (as will be explained later). Requesting data from the *SWM* is achieved by performing a semantic translation which generates the requested property based on the raw state of an object within the game engine. To give an example, for an agent with the required ability to navigate through doors, one could define an ontological object class *Door* with a property *status* specifying whether the door is open or closed. At runtime, upon an agent request for this property, a translation is performed calculating the status of the door based on its geometrical position relative to its surrounding geometry. Since every game object that should be perceivable by agents is required to register itself within CIGA and specify the ontological class it represents, CIGA is able to relate concepts in the *SWM* to objects in the game engine.

4.2 Embodied Agent Layer

CIGA's *Embodied Agent Layer* regulates the sensing process of an agent using one or more virtual sensors. On the game engine side, it can access the virtual environment to perform sensory queries. Within CIGA, it can access the *Semantic World Model* for generating sensory information. On the MAS side, it interfaces with a *Perception Module* assumed to be available in the MAS where received sensory information is processed and perception control messages flow vice-versa. Next we look into the individual components in more detail.

Sensory Processor. This is where the agent's update cycle for sensing is controlled from. First individual sensors are given a chance to update, generating and storing sensory information inside a buffer (the *Object Sign Buffer*). Next, at the end of an update cycle, data collected inside this buffer is sent as individual percepts towards the MAS side.

Interest Manager. This component is part of an agent's sensory processor and contains an agent's interest in environmental data. It manages *subscriptions* on certain information from the environment for which sensory information should be generated. These subscriptions reference ontological concepts defined within the application's *Ontology Model* and can be adopted or dropped from the MAS side. This subscription-based approach allows an agent to have full control over the flow of sensory information it generates. A common type of subscription concerns a frequency-based subscription for an object property. Its states that an agent has an interest to receive some property (e.g. *location*) of a certain object class (e.g. *Human*) at a fixed rate (e.g. 5 times per second). More information on this subscription-based technique can be found in [17].

Virtual Sensors. Sensors are based on an abstract class from which concrete sensors can be created for a certain modality (e.g. visual, auditory). The execution of a sensor is based on a *sensing algorithm* which is performed at each sense cycle. Although the specific algorithm may differ for each sensor, it typically involves (1) accessing the virtual environment in the game engine to find out what elements falls within the sensor's sensory range, (2) accessing the *Interest Manager* to consult the currently active subscriptions, (3) accessing the *Semantic World Model* to request specific semantic object properties and (4) accessing the *Object Sign Buffer* to store the requested data as *object signs*. In this paper we will only consider visual sensing for retrieving object properties since this is typically both the most common and most computationally expensive part of a perception system. However, optimization approaches proposed in this paper are not specific for visual sensing and the same concepts can be utilized for other sensor modalities.

4.3 Subscription-Based Filtering

As described earlier, the representation of sensory information is in accordance with the concepts available in the semantic world model (SWM). In subscription-based filtering we employ this model to define *subscriptions* that represent an agent's perceptual interest towards its environment. For example, an agent can subscribe to specific properties from one or more specific object classes. Here the hierarchical nature of objects defined in the *SWM* is taken into account: i.e. an agent having a subscription for the location of all physical objects indirectly has a subscription for the location of all object classes defined as a subclass of a physical object.

A subscription specification consists of two mandatory elements that must be specified. The first element represents the nature of the semantic interest, referencing elements from the *SWM* (e.g. an object property). The second element refers to an update policy. An update policy specifies at what times or how often there is a demand for the semantic interest. Different types of update policies are available: a *one-time* subscription (desires information once), a *frequency-based* subscription (desires information at a certain frequency) and a *value-change* subscription (desires information when it changes). Finally as an optional feature, one can provide a conditional source which specifies a specific source object from which information is desired. Employing above features for subscriptions provides a powerful mechanism to specify desirable sensory input for an agent, only limited by the richness of the semantic concepts defined in the *SWM*.

More information on subscription-based filtering and an evaluation of its performance gains can be found in [17]. In the remainder of this work we propose optimization approaches for an agent's perception process within CIGA.

5 Towards Optimizations

In previous experiments [17] it was seen that although the computational time required for agent sensing could be decreased based on the introduction of subscription-based filtering (hereby eliminating the need to create and send irrelevant sensory information), still, the system formed a bottleneck caused by the required processing of the visual sensor in highly populated environments. In this section optimizations to increase efficiency are investigated.

5.1 Optimizing Visual Sensing

The sensing algorithm of an agent's visual sensor can be broken down into two phases: a *physical* and a *cognitive* processing phase. The former determines what

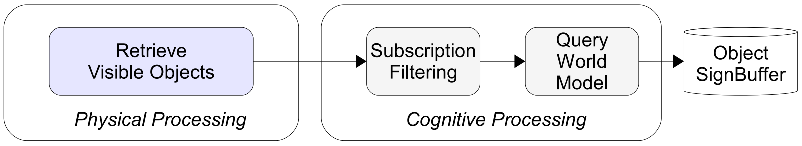


Fig. 2. Basic Sensory Algorithm

objects in the environment are *observable* for an agent and acts a *physical filter*. The latter is a *cognitive filter* and generates sensory information only for those objects that (1) pass the physical filter and (2) contain properties an agent is subscribed to. This process is shown in Figure 2.

Physical processing must be executed in a game engine and builds a list of environment objects observable to agent's visual sensor which means they are (1) within visual range and (2) not obscured by other objects. A commonly used implementation is to filter objects first by eliminating those which fall outside a certain scope (e.g. a viewing frustum) and second performing *line-of-sight (LoS) checks* for remaining objects (e.g. using ray casting techniques). In this context, a lot of irrelevant processing may take place caused by unnecessary *LoS checks* being performed: there is no need to perform *LoS checks* for objects for which no subscription matches. As an optimization, we propose an alternative visual sensing algorithm as shown in Figure 3.

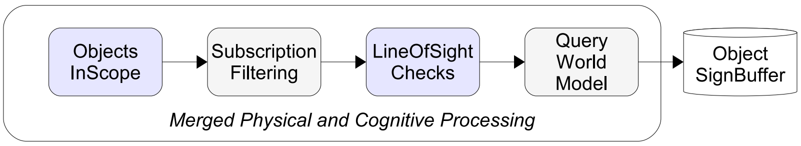


Fig. 3. Optimized Sensory Algorithm

This algorithm merges physical and cognitive processing by dividing the former into two separate sub-processes where retrieved objects within scope are first filtered on available subscriptions before considering whether the object in scope is actually visible. Consider an agent presented with 10 boxes and 10 spheres within visual range which has a subscription on the *color* property of boxes. In case of the optimized algorithm, only 10 *LoS checks* have to be performed (only for boxes) instead of 20 when using the basic algorithm. A disadvantage of the optimized algorithm is that we loose a clear conceptual separation between physical and cognitive processing. Still, it allows an agent to cope with more complex environments by filtering environment objects in an earlier stage of perception, based on an agent's demand for sensory information.

5.2 Caching the Semantic World Model

As described in Section 4.1, an agent can request an object property from the *Semantic World Model* (*SWM*) when the corresponding object has passed the

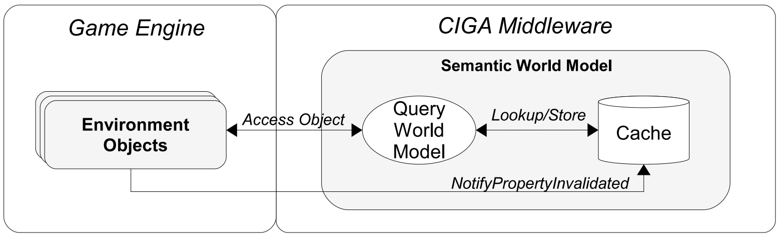


Fig. 4. Semantic World Model with Caching

physical and cognitive filters. Such a request is handled by performing a semantic translation based on the object's current state to generate ontologically-grounded data used as sensory information. Now considering the required computations, it can be seen cumbersome to perform translations for consecutive requests when the corresponding object's status remained unchanged. Such redundant operations can be prevented by caching previously obtained results. A design approach for employing caching is shown in Figure 4.

In this approach the environment objects themselves have the responsibility of notifying the *SWM* about a change in their state related to one or more semantic properties. A *Cache* is introduced for storing recently performed semantic translations where a cache entry consists of an object identifier, a property name, a value for this property (result of a translation) and an *invalidated* field. Whenever the cache receives an invalidation notification from an environment object, the *invalidated* field of the corresponding cache entry will be set to true, meaning the currently cached value is out-of-sync with the actual state of the corresponding object.

A request for an object property from the *SWM* is handled by the procedure defined as *QueryWorldModel*. It will search the cache for a corresponding entry for this property. If found and not invalidated, the property's value in the cache is returned. If invalidated or no entry has been found at all, a new semantic translation is required and the property's value is created based on the object's state in the game engine. The newly created value will be put in the cache (either by updating or inserting a new cache entry) and marked as not-invalidated.

The advantage of the proposed design is that semantic translations are performed only when necessary caused by an object's state change. The downside is the increased burden for a programmer to keep track of changes, relating them to the object's semantic properties and correspondingly notifying the *SWM*. Further, bi-directional dependencies are created between a game engine and CIGA (and thus a MAS), resulting in a more tightly coupled software connection.

6 Experimental Evaluations

Evaluating the optimizations proposed in the previous section, we report on several experiments that were conducted to analyze the impact on an agent's sensing process. We start by providing implementation details of the system that was used after which experimental evaluations are presented.

6.1 System Implementation

As seen in Figure 1, a system implementation involves three architectural components, namely a game engine, a multiagent system and the CIGA middleware. As for the game engine, a company's in-house developed C++-based game engine was employed in which human-like characters have been designed with basic abilities for sensing and acting. Although no well-known game engine was employed, the used engine includes common industry-standard components that can be expected nowadays from a state-of-the-art game engine (e.g. scene management, a physics engine and an animation engine).

The multiagent system has been custom built in Java. The reason not to use a standard agent platform (e.g. Jason, Jadex or JACK) is that our experiments do not require any advanced decision-making or planning. Further, the environment interface that such platforms offer is often fairly primitive where there are no facilities for agents to communicate messages to their embodiment (except for using actions). Last, since the perception module on the MAS side had to be programmed separately to interface with the middleware, we decided to program the complete agent directly in Java as well. This does *not* mean that standard agent platforms cannot be used, but rather that the overhead in this case did not warrant the advantages.

Finally, the CIGA middleware employs a distributed design interfacing with both the game engine and the MAS in their own language. Internally, TCP/IP is used for inter-process communication. For our evaluation domain, an ontology was created for the *Semantic World Model* in Protégé. This ontology can be loaded into the middleware and referenced from both the game engine and the MAS. More technical design details of how the middleware interfaces with the game engine and the MAS fall outside the scope of this paper, but can be found in [18]. Next we elaborate on the processes related to agent sensing within the proposed framework from Figure 1.

6.2 Evaluation Details

For a better understanding of the scope of computational processing within an agent's sense cycle, Figure 5 shows a process hierarchy of the relevant processes involved. Note that we only consider an agent to have a visual sensor. The left sub-tree denotes the process of updating the visual sensor whose sub-processes have been explained in Section 5.1. The right sub-tree denotes transporting sensory information stored in a buffer towards the agent's perception module on the MAS side using the IPC. It consists of (1) preparing sensory information for transport through serialization and (2) transporting the result using TCP/IP socket communication.

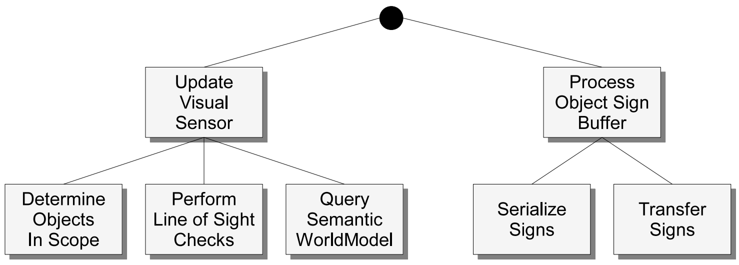

Update Visual Sensor
Process Object Sign Buffer
Determine Objects In Scope
Perform Line of Sight Checks
Query Semantic WorldModel
Serialize Signs
Transfer Signs

Fig. 5. Sensing Process Hierarchy

A profiler has been developed which for each process in the hierarchy can measure the computational time spent on a particular process during one sense cycle. External influences may affect these measurements (e.g. concurrently running background processes). For more reliable results, averages are calculated over measurements from several separate sessions of at least 100 update cycles. Within the application, scenarios can be designed where the environment can be populated with different types of geometric shapes (namely boxes and spheres) which can vary along several dimensions (e.g. location, color). A single agent is included which can be given subscriptions on one or more dimensions of the objects. Figure 6 gives an impression of the kind of scenarios that were used.

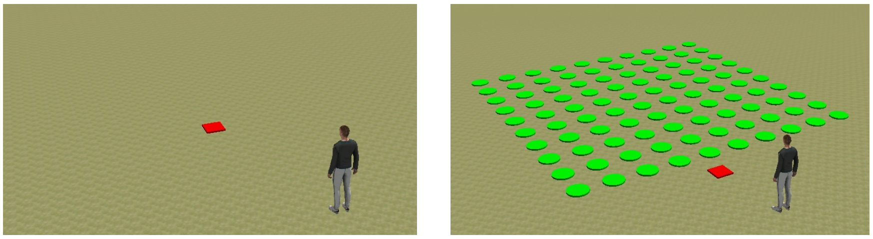


Fig. 6. Scenario Impressions

To give an example of the actual data communicated between an agent in a MAS and its embodiment in the game engine below illustrates an agent's message to subscribe to an object property and a corresponding percept related to the subscription.

```
(sender=john)(receiver=john)(time=153.0)
(content=CommandSubscribeInterest
    (interestID:1)
    (interest:<InterestObjectAttribute
        (interestFrequency:5)
        (objectClassHandle:Box)
        (objectAttributeID:color)>
    )
)

(sender=john)(receiver=john)(time=163.0)
(content=ObjectPercept
    (objectID:box6)
    (object:<Cube
        (color:red)>
    )
)
```

Below we will continue with a description of the experimental evaluations.

6.3 Visual Sensing Analysis

This first experiment involves a comparison between the two visual sensing algorithms discussed in Section 5.1, getting insight into the possible performance gain of using the optimized algorithm. The experiment includes several scenarios where the environment contains a single box and zero or more spheres (varied between 5 and 100 across consecutive scenarios). The agent is given a subscription on the location property of *boxes*. By increasing the amount of spheres, the number of objects the agent will not be interested in is increased where efficiency can be gained by eliminating irrelevant LoS checks (as done by the optimized algorithm). Each scenario will be run twice, once for the default visual sensing algorithm and once for the optimized one. Results can be seen in Figure 7, plotting the measurements recorded for the process *UpdateVisualSensor*.

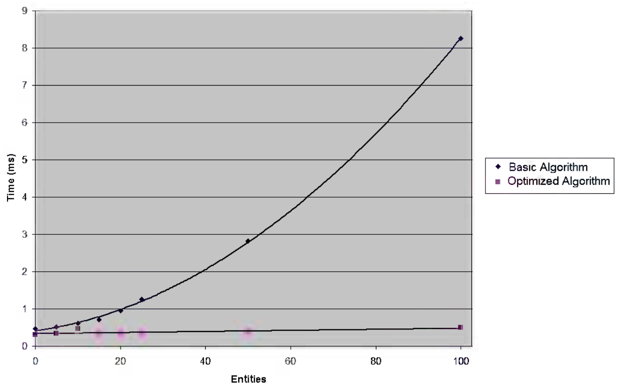


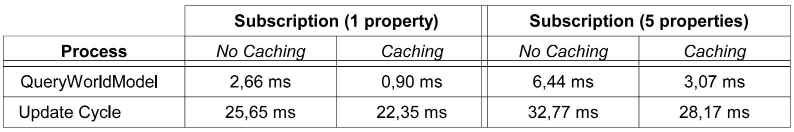
Fig. 7. Experimental results for the process *UpdateVisualSensor*

In the figure, the horizontal axis resembles the different scenarios, increasing the number of entities in the environment (in this case *spheres*). The vertical axis illustrates the computational time for an agent spent on the process *UpdateVisualSensor* for each scenario. This process is shown in the process hierarchy in Figure 5 and covers all processes involved in an agent's sensing algorithm (e.g. see Figure 3).

One can see that for each consecutive scenario, the basic algorithm takes substantially more computational time whereas the time measured for the optimized algorithm stays stable throughout the different scenarios. The difference in computational time is fully caused by the different number of *LoS checks* that have to be performed. In the optimized algorithm, only one check has to be performed throughout all scenarios (for the single box) whereas in the basic algorithm this ranges from 6 to 101 checks. Of course, since in the optimized algorithm the same kind of calculations is performed but only less of them, the results may not be surprising. Still, results do show that performing a *LoS check* is a computationally expensive operation and one can gain a lot of performance from a relatively simple mechanism like subscription-based filtering, making sure that only those checks are performed that are actually relevant to an agent. Concluding, using the optimized algorithm, a substantial performance gain can be achieved, especially in situations where the number of environmental objects for which an agent has no cognitive interest in is increased.

6.4 Semantic Cache Analysis

With a second experiment, we analyze the possible performance gain of using caching as described in Section 5.2. A comparison is made between the computational times required for retrieving cached object properties versus retrieving those properties through semantic translation. Two scenarios are performed in which the agent is presented with 150 boxes visually perceivable. In the first scenario the agent is subscribed to a single property of a box whereas in the second scenario the agent is subscribed to 5 different properties. Both scenarios are run twice, once with caching enabled and once without caching. The state of all objects remains unchanged through the scenarios, resulting in positive cache hits in case caching is enabled. The results are illustrated in table 8, showing computational times for querying the world model and the full update cycle. The former is shown in Figure 5 as the process *QueryWorldModel*. A full update cycle entails the full process hierarchy from this figure (i.e. the root of the hierarchy).



	Subscription (1 property)		Subscription (5 properties)	
Process	*No Caching*	*Caching*	*No Caching*	*Caching*
QueryWorldModel	2,66 ms	0,90 ms	6,44 ms	3,07 ms
Update Cycle	25,65 ms	22,35 ms	32,77 ms	28,17 ms

Fig. 8. Experimental Results

Analyzing the results, we can see that for a 1-property subscription (querying 150 properties) the process *QueryWorldModel* performs 2.96 times better with caching than without. With a 5-property subscription (querying 750 properties), it performs 2.2 times better with caching enabled. This difference can be explained by the fact that the time required to perform a semantic translation can differ between different semantic properties. Generally, a property defined at a higher abstraction level from the object's raw state requires a more complex translation rule, resulting in an increased benefit for employing caching.

Considering the impact of caching on the computational time required for a full sense cycle, for a 1-property subscription the gain is about 13% whereas for a 5-property subscription about 14%. Putting these numbers into perspective requires a full analysis of all processes involved in an update cycle as will be given in the next section. We can conclude that the use of caching can increase the performance of querying the world model, though the gain is relatively small in scope of the overall update cycle. Further, the gain is not always the same and depends on the complexity of a semantic translation to be made. Last, one has to remember that such a gain would not be achieved for objects that become invalidated every game cycle (resulting in invalidated cache entries).

In this experiment we have considered the most 'optimistic' situation in which the environment is fully static. Here, a performance gain was measured around 13%-14%. In more dynamic environments, one can expect less gain in performance.

7 Framework Performance Analysis

In this section we provide a performance analysis of the perception framework within a scenario in which an agent increasingly generates more sensory information. It is aimed to (1) analyze the computational complexity of the significant processes within the framework and (2) to investigate the relative computational times between those processes in order to identify a bottleneck in the system.

To accomplish this, an experiment was conducted in which an agent was provided with a subscription on the location property of boxes while the number of boxes within the environment increases across consecutive scenarios. Important to note is that here one will not benefit from the optimized visual sensing algorithm from Section 5.1, since only objects are used for which sensory information must be generated (thus requiring LoS checks). Further, the framework was configured to not employ caching within the *Semantic World Model*, since this design choice is highly application dependent. Also, by not employing caching, we can treat the experiment results as if *dynamic* objects were used: whenever required by the agent's subscription, translation rules are being performed independent of whether or not the objects are static or dynamic. The results from the experiment are shown in Figure 9. It covers the process hierarchy from Figure 5 where the left side of the figure includes the sub-processes for *UpdateVisualSensor* and the right side includes the sub processes for *ProcessObjectSignBuffer*.

Based on these results, we can see that the *LoS checks* within the visual sensor becomes the first bottleneck because of its non-linear nature. The process *QueryWorldModel* is not negligible and one may consider employing caching to further decrease its impact. Considering buffer processing, this also takes a significant amount of time (approx. 50% of the time of the overall sense cycle

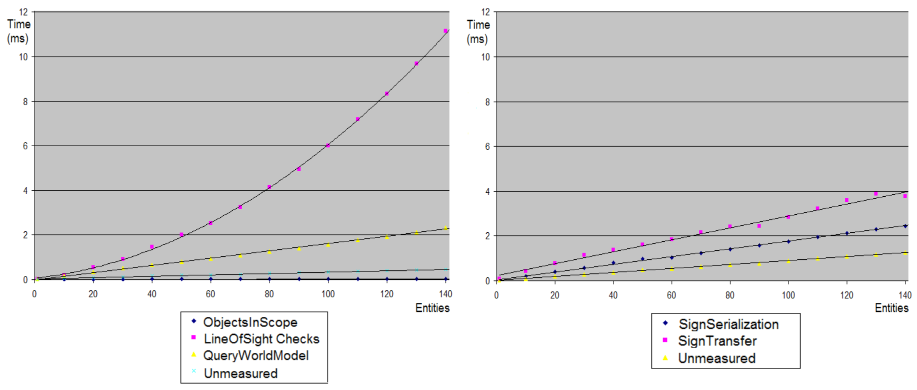


Fig. 9. Experimental Results for leaf processes of the process hierarchy

when retrieving sensory information for 150 objects). Here, both serialization and transfer of sensory information contribute significantly. Since we have used a human-readable format for sensory information, we expect one can decrease this percentage when considering more efficient serialization techniques (with better performance and a smaller message size).

Although we have only used a single agent in this experiment, using more agents would not have an effect on the performance measurements of an *individual* agent. Only when caching is employed, agents are able to share the results of semantic translations made by the *Semantic World Model*. Though, we have already seen that this performance gain would be fairly small compared to the computational time of the *LoS checks*.

8 Conclusion

In this paper we have proposed optimizations and conducted an analysis of an agent perception framework employed within CIGA, a middleware mediating between a game engine and a MAS covering the mind-body interface for agents embodied in a real-time virtual environment.

Concerning the proposed optimizations, we conclude the following. First of all, one can gain much performance by having an agent's cognitive interest in environmental objects control its physical sensing process in a game engine. Objects for which no sensory information is requested do not require a full visual confirmation and this is eliminated when employing the optimized visual sensing algorithm. This allows an agent to cope with more complex environments by ignoring 'uninteresting objects'. On the downside, one requires access to more specialized functionality from the game engine (e.g. LoS queries) which results in a tighter coupling between a game engine and CIGA (and thus a MAS).

Second, considering the design approach of caching within a semantic world model, one will gain performance in situations where the state of a perceived object remains unchanged between consecutive observations (eliminating possibly complex semantic translations). Performance gains were measured at about 13-14% for a sense cycle. Looking from a design perspective, the use of caching imposes specific requirements for a game engine: environmental objects are required to notify the semantic world model within CIGA during state changes. Here, a tradeoff is in place between efficiency and design complexity.

Last, concluding on the performance analysis of the overall framework, a bottleneck in the system can emerge when an agent requires many visual confirmations for objects in the environment (i.e. for which a cognitive interest exists). Thus, although an agent may be able to deal with heavily populated environments (by employing the optimized sensing algorithm), one still has to be careful in controlling an agent's cognitive interest using the subscription-based filtering mechanism provided by CIGA, as to avoid performance issues.

Putting our contribution in a broader scope, we have seen that performance issues concerning perception for agents embodied in a real-time environment have been recognized before. Although optimization approaches have been presented within the scope of CIGA, conceptually, similar approaches can be applied to other architectures. These approaches relate to (1) centrally managing a semantic world model and (2) a subscription-based percept filtering mechanism, employing the concepts defined in the semantic world model.

As for future work, it was seen that the focus should be on further optimization approaches for determining the environmental elements that agents should physically be able to see within a virtual environment. One can think of approaches to share sensory computations performed in the game engine between agents, realized by some central mechanism.

References

1. Adobbati, R., Marshall, A.N., Scholer, A., Tejada, S.: Gamebots: A 3d virtual world test-bed for multi-agent research. In: Proceedings of the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS (2001)
2. Behrens, T., Hindriks, K., Dix, J.: Towards an environment interface standard for agent platforms. Annals of Mathematics and Artificial Intelligence, 1–35 (2010)
3. Chang, P.H.-M., Chen, K.-T., Chien, Y.-H., Kao, E., Soo, V.-W.: From Reality to Mind: A Cognitive Middle Layer of Environment Concepts for Believable Agents. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2004. LNCS (LNAI), vol. 3374, pp. 57–73. Springer, Heidelberg (2005)
4. Endsley, M.: Designing for Situation Awareness: An Approach to User-Centered Design. Taylor & Francis (2003)
5. Evertsz, R., Pedrotti, M., Busetta, P., Acar, H., Ritter, F.: Populating VBS2 with realistic virtual actors. In: Proceedings of the 18th Conference on Behavior Representation in Modeling and Simulation, pp. 1–8 (2009)
6. Gemrot, J., Brom, C., Plch, T.: A periphery of pogamut: From bots to agents and back again. In: Dignum, F. (ed.) Agents for Games and Simulations II. LNCS, vol. 6525, pp. 19–37. Springer, Heidelberg (2011)
7. Grimaldo, F., Lozano, M., Barber, F., Vigueras, G.: Simulating socially intelligent agents in semantic virtual environments. Knowledge Engineering Review 23, 369–388 (2008)

8. Herrero, P., Greenhalgh, C., Antonio, A.: Modelling the sensory abilities of intelligent virtual agents. Autonomous Agents and Multi-Agent Systems 11(3), 361–385 (2005)
9. Leonard, T.: Building an AI Sensory System: Examining the design of Thief: The Dark Project. In: Game Developers Conference (GDC 2003) (2003)
10. Maes, P.: Modeling adaptive autonomous agents. Artificial Life 1(1-2), 135–162 (1994)
11. Peters, C., Castellano, G., Rehm, M., André, E., Raouzaiou, A., Rapantzikos, K., Karpouzis, K., Volpe, G., Camurri, A., Vasalou, A.: Fundamentals of agent perception and attention modelling. In: Cowie, R., Pelachaud, C., Petta, P. (eds.) Emotion-Oriented Systems, pp. 293–319. Cognitive Technologies (2011)
12. Reynolds, C.: Interaction with groups of autonomous characters. In: Proceedings of Game Developers Conference, San Fransisco, CA, pp. 449–460. CMP Game Media Group (2000)
13. Riedl, M.O.: Towards Integrating AI Story Controllers and Game Engines: Reconciling World State Representations. In: Proceedings of the 2005 IJCAI Workshop on Reasoning, Representation and Learning in Computer Games (2005)
14. So, R., Sonenberg, L.: The roles of active perception in intelligent agent systems. In: Lukose, D., Shi, Z. (eds.) PRIMA 2005. LNCS, vol. 4078, pp. 139–152. Springer, Heidelberg (2009)
15. Hindriks, K.V., van Riemsdijk, B., Behrens, T., Korstanje, R., Kraayenbrink, N., Pasman, W., de Rijk, L.: Unreal Goal Bots. In: Dignum, F. (ed.) Agents for Games and Simulations II. LNCS, vol. 6525, pp. 1–18. Springer, Heidelberg (2011)
16. van der Vecht, B.: Adjustable Autonomy: Controling Influences on Decision Making. PhD thesis, University of Utrecht (2009)
17. van Oijen, J., Dignum, F.: Scalable Perception for BDI-Agents Embodied in Virtual Environments. In: Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, WI-IAT 2011, vol. 02, pp. 46–53. IEEE Computer Society, Washington, DC (2011)
18. van Oijen, J., Vanhée, L., Dignum, F.: CIGA: A Middleware for Intelligent Agents in Virtual Environments. In: Beer, M., Brom, C., Dignum, F., Soo, V.-W. (eds.) AEGS 2011. LNCS, vol. 7471, pp. 22–37. Springer, Heidelberg (2012)
19. Weyns, D., Steegmans, E., Holvoet, T.: Towards active perception in situated multi-agent systems. Applied Artificial Intelligence 18(9-10), 867–883 (2004)

Ambient Intelligence with INGENIAS

Jorge J. Gómez-Sanz, José M. Fernández-de-Alba, and Rubén Fuentes-Fernández

GRASIA Research Group,
Universidad Complutense de Madrid,
Avda. Complutense, 28040 Madrid, Spain
{jjgomez,jmfernandezdealba,ruben}@fdi.ucm.es

Abstract. This paper introduces advances made in the INGENIAS methodology to deal with Ambient Intelligence (AmI). The work considers the particular features of AmI systems and how an agent-oriented methodology can help to address their development. Being INGENIAS a model-driven methodology, a first step has been to compare the concepts used in INGENIAS and those required by AmI developments. This analysis identifies required changes to be applied in the INGENIAS metamodel. A case study on a tracking system to locate people illustrates this discussion.

1 Introduction

Ambient Intelligence (AmI) is a generic name given to the integration of different information technologies that enables artificial environments to react and advance human actions. Quoting Aarts and Wichert [1] *Ambient Intelligence is about sensitive, adaptive electronic environments that respond to the actions of persons and objects and cater for their needs.* The presence of heterogeneous sensors and actuators, and intelligence (e.g. in form of adaptiveness or awareness) in AmI systems makes their development very challenging for engineers.

Looking for approaches that facilitate this development, some authors have considered the use of Agent-Oriented Software Engineering (AOSE) [12]. Agent technology already offers some of the features required for AmI: agents are intelligent components, heterogeneous and inherently distributed, able to manipulate other artefacts and collaborate to achieve higher-level goals. This line of work is suitable for research on AmI focused on intelligence aspects. Nevertheless, there are other works pursuing complex hardware deployments and lightweight reasoning apparatus where agent technology is harder to fit.

Despite the potential synergies, the integration of AmI and AOSE is still quite preliminary. Sadri [12] makes an extensive review of agent systems applied in AmI. Most of them are introduced as ad-hoc systems, i.e. tailored for specific domains and developed without following a general process. A possible reason for this situation, according to Sadri [12], is the trade-off between using generic approaches and their level of support. A general approach reduces learning and tool acquisition costs, as it can be applied in multiples contexts and projects.

However, these approaches usually provide little effective guidelines due to heterogeneity and constraints in AmI systems. A relevant example of this are the issues with their hardware.

Sensors and actuators have limited computational capabilities, so not any software can run into them. Moreover, these devices do not conform to unique standards regarding aspects such as programming, communications or discovery. For instance, software running in different ambient devices may find difficulties to inter-operate. The problem lies frequently in the use of different standards that were not designed with a general purpose interoperability in mind. As an example, let us consider Ambient Assisted Living (AAL), the application of AmI for assisting people. The number of standards involved in this domain is hard to account [6]. Only for communications, systems frequently use the Open Service Gateway Initiative (OSGi), the Common Object Request Broker Architecture (CORBA), web services, and Universal Plug and Play (UPnP), to cite some. There are also more specific standards in some domains. For instance, the health sector uses among others: the ISO EN 13606 Electronic Health Record Communication (EHRcom) for semantically interoperable exchange between electronic health files; Health Level Seven (HL7) for data exchange between organizations; or the ISO/IEEE 11073 for data exchange of medical sensor signals and vital signs from patient-related devices. So, until interoperable AmI hardware is widely available, software will have to deal with this heterogeneity.

The goal of this paper is to achieve a greater integration between AmI and AOSE. The main reason is making the development more systematic. With the support of an agent oriented development process which takes into account needs and limitations of AmI, necessarily the development will progress more steadily: there will be tools, identified activities that will drive the development, and notation/vocabulary to express the different products. This first work focuses in the vocabulary and notation and pursues enhancing an AOSE modeling language to capture AmI concerns. There are many existing AmI and AOSE alternatives. Nevertheless, as it has been pointed out before, synergies between both have not been studied, yet.

A possible proof-of-concept integration is being addressed within the Soci-AAL project (TIN2011-28335-C02-01). This project intends to reduce development costs can be while keeping technology independence through the use of the model-driven development paradigm. The method used in the project consists on capturing the commonalities of AmI into a meta-model, and devising ways in which this meta-model can be realized into different target platforms. In our case, the basis of this meta-model is the agent paradigm. The control of AmI systems will be embodied into software agents built following an agent-oriented methodology, INGENIAS [11] in this case. The reason for choosing using INGENIAS is its support for the model driven paradigm. Unlike other methodologies, INGENIAS was born with model driven tools adapted to a concrete meta-model, which written using XML format. Also, a researcher could evolve INGENIAS meta-model while keeping its associated tools updated. This was possible because of the INGENME framework (`http://ingenme.sf.net`) that enables the automatic creation of self-contained visual editors using an XML description of the meta-model. Other methodologies have just recently acquired this capability [8] by migrating their facilities to the Eclipse Modelling Framework.

In order to ground this discussion, this paper considers the FAERIE [4,5] AmI platform, which is an ongoing development in our research group. This initial experience will provide us the basis to study other platforms [2,12], and determine what improvements can bring the use of INGENIAS to develop technology independent AmI systems. For us, the *context* will be something our agents connect to. Once connected, they receive or send information. Devices connect as well to the *context* directly, enabling interaction through shared pieces of information, or through agents, developing wrappers with an API our agents can handle.

The paper is structured as follows. First, there is an introduction to the INGENIAS meta-model in Sect. 2. Next, Sect. 3 explains the integration of AmI concepts into this meta-model. It is based on the use of the *context* concept to model different parts of the system, including its deployment. Section 5 introduces the FAERIE AmI software and shows how the generic meta-model can be specialized for it. A case study on people tracking in Sect. 6 illustrates the application of this new meta-model to develop AmI systems. The two final sections contextualize this research with related work (see Sect.7 and discuss some conclusions on it (see Sect. 8).

2 A Fragment of INGENIAS Metamodel and Integration Proposal

The integration of an AmI solution into INGENIAS requires introducing first the relevant key elements of the INGENIAS meta-model. Figure 1 presents a selection of actual elements of that meta-model that captures most of the cycle for information processing.

The meta-model representation follows the GOPRR [9] approach. It considers entities that can be either a *Graph* (i.e. a diagram), an *Object* (i.e. a node in the diagram), a *Property* (i.e. an attribute), a *Relationship* (i.e. an edge), or a *Role* (i.e. a connection between an *Object* and a *Relationship*). This information is expressed in Fig. 1 using stereotypes. This meta-model is then processed by the INGENME `http://ingenme.sf.net` framework to produce tailored visual editors.

Regarding the cycle followed by INGENIAS agents, it is based on Newell's *rationality principle* [10]. This principle states that an agent chooses for execution those actions aimed at satisfying its goals. Hence, the agent behavior can be described in terms of the goals it pursues, the activities it can use to achieve those goals, and how it handles the information.

Figure 1 describes this cycle introducing several concepts. An *agent* pursues a *goal*. This *goal* is in fact a *mental entity* whose purpose is controlling the *agent*. The *goal* is satisfied when a *task* is executed and some concrete evidences are produced. The *task* uses some *applications*, which represent the actuators of the agent. They are conceived as software wrappers that adapt whatever elements

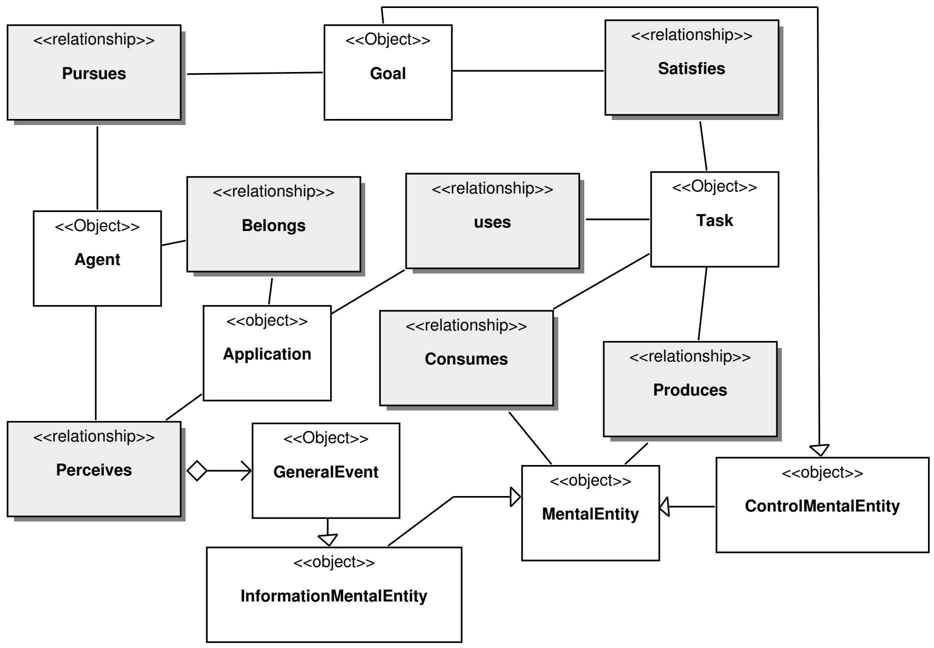


Fig. 1. INGENIAS metamodel fragment with elements participating into event processing

exist in the environment to an API that is accessible by the *task*. Hence, the *task* is able to alter the environment of the agent and stimulate the production of *general events*. These events are attended by the *agent* and treated as a regular *mental entity*, e.g. they can be used to trigger other *tasks* or to consider some *goals* as satisfied.

3 Extending the INGENIAS Metamodel

The goal of this work is to integrate AmI concepts into the MAS modeling language of INGENIAS. For doing this, it is necessary to understand the role of the *context* in AmI, which appears regularly in the AmI literature. The experience introduced in this paper focuses on how the *context* appears in Bolchini [2], which surveys different representations of it. The *context* is described in [2] as an *active process dealing with the way humans weave their experience within their whole environment, to give it meaning*. The *context* is a representation of different pieces of information from the environment, the systems and activities. It includes attributes dealing, for instance, with space (location of elements), time (when each piece of information was produced) and relationships. There are also rules describing reasoning on these attributes, for instance how to update calculated attributes. This information is used by people and applications.

The implementation of the context can be centralized (chosen in most cases) or decentralized. This *context* is generally a passive element, so there are entities making it change, but always regarding specific constraints. Most of the approaches with context modeling and implementation are based on blackboard approaches.

A *context* is not expected to be static. For instance, a user may carry its personal *context* with him, perhaps in form of wearable devices or mobile phones. Such *personal context* changes in several ways. Changes as wearable devices are put on or discarded. It changes as well as *context* information is beeing updated. Also, since the user moves himself, he can meet other *contexts* along the way with which interaction may be necessary. Besides, actors involved into other *contexts* may request interacting with ours, for instance, to offer customized advertising or to build a new service for other *contexts*.

So, a *context* can be understood as a place where information is stored and shared. Access to this store is regulated since there are strong concerns about privacy. Actions within the *context* would be carried out by the actors accessing to it, though it is possible the *context* itself has some built-in operations. In AmI, the capabilities available in a *context* are dependent on the kind and number of sensors and actuators that are already associated to the context.

Starting with the integration, we assume that active entities in the *context*, the ones connecting and disconnecting, the ones performing concrete actions and accessing the information, they are agents. In INGENIAS *agents* are the active entities. It makes sense that they are the responsible of interacting with the *context*. This *context* is a container of information capable of event notification and offering some pre-arranged operations. The association of an INGENIAS *agent* with the context is called *binding*, and one agent can be *bound* to several *contexts*. Figure 2 represents with a state machine the life-cycle that links an agent and a context.

The cycle starts when the agent receives an event indicating there is a context available. This event may be generated by some external component following a given policy (e.g. periodically searching for contexts), or triggered purposely by the agent through some specific system request.

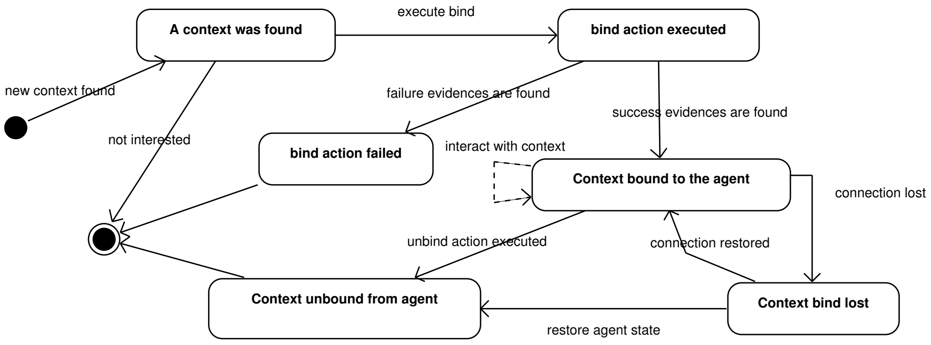


Fig. 2. Lyfecicle of the context to agent association

When the context is found, the agent decides if there is interest in joining in. If there is, the agent launches a binding action. Since the communication paradigm for agents is inherently asynchronous, after launching the binding action, the agent does other tasks, and it processes the obtained results when available. The collected evidences by the agent may indicate the success or failure of the action. If the binding has not failed, the agent can interact with the context from that moment and on. While this binding exists, the agent may loss the connection with the context, so information exchange actions may suffer some delays and the participants need to reestablish the connection. The agent to context binding ends if the connection is not restored or the agent purposely unbinds. The inclusion of this cycle in the INGENIAS meta-model affects several elements. These ideas are captured in the metamodel from figure 3.

First, the AmI environment is represented with a *context* entity. This entity is associated to a *context model* and represents a set of pieces of information that are coherently grouped. One context can have several associated context models, which is necessary to address the heterogeneity of AmI environment. The *context* identifies the elements the agent will interact with. The *context model* provides an information model of the kind of information to be found within the *context.* This is necessary to define agent's behavior, as it is described in terms of tasks and the information they may have as input and produce. *Context modeling* is challenging since there are many possible information structure used at this moment in the literature. At this point, the extension of the meta-model just considers some context modeling may be required and we provide a concrete context modeling solution for FAERIE, as section 5 presents.

The interaction between the context and the agent is simplified with an updates/notifies relationship. An *update* implies an *agent* pushing information into the *context. Notify* is the reverse operation, because it is the *context* the one pushing information to the *agent.* It serves to provide high level representations of the problem, as it is done in figure 7. The agent will be aware of the existence of contexts either by looking for one actively or assuming there is a mechanism that enables an agent to be notified when one context appears. Interaction with the context is delegated to specific task types, namely *Bind Context Task*, *Release Context Task*, and *Context Use Task.* Using a *Bind Context Task* (see figure 3), the agent knows which contexts are available and may request to gain access to one. When the agent decides to leave a context, it will execute a *ReleaseContext-Task.* It may also be the case that the connection with the *context* is lost, as when having an agent in a mobile device that gets out of the reach of the context device. In these cases, a *Context Lost Event* would be produced. The *Context Use Task* exists to deal with general interaction with a context once the agent is inside. For instance, to request listening to changes in one entity or to perform changes in another.

As stated in the introduction (see Sect. 1), our work also relies in the *context* concept to describe the system deployment. Figure 4 explains this role. The INGENIAS modeling language includes *deployment packages* to specify collections of agent instances created using concrete pieces of information. In runtime, each

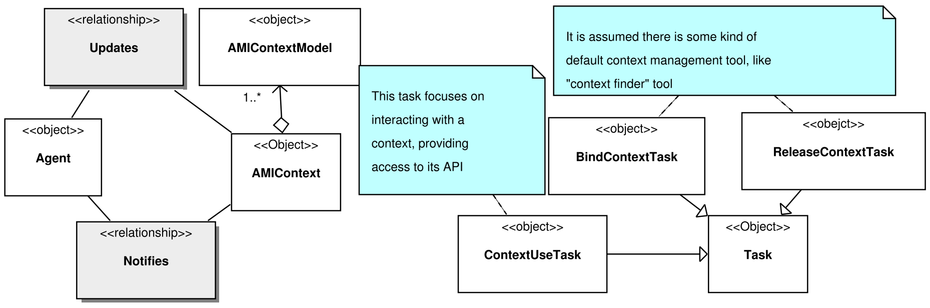


Fig. 3. INGENIAS metamodel fragment of elements participating into event processing

deployment package is run in a separated Java Virtual Machine. The *Deployment Unit By Type* allows creating a number of instances of one agent type without parameterizing it. The *DeploymentUnitByTypeEnumInitMS* creates a number of agent instances using the pieces of information declared in a mental state. The *DeploymentUnitByTypeEnumMSEntity* does something similar, but the pieces of information for the initialization are listed individually, and not as part of an agent mental state. For AmI, a new *deployment unit* type is added, the *Deployment Agent with Context*. It includes all the information that one agent needs to access to a *Context* it is bound. It assumes a context is launched as well within the deployment package. The *context* is associated to several *Context Models* (see figure 3), so it is required as well to determine how the model is instantiated. This is the role of the *Context Model Instantiation* entity which uses a facility of INGENIAS metamodel to express at the modeling level instances of the model. The name of this facility is *Mental Instance Specification* and is suited for specifying the attribute values of any instance extending the *Mental Entity*, in the INGENIAS meta-model. With these elements, it is possible to create a JVM where a selected group of agents, probably parameterized, coexist with a context, which contains the pieces of information determined in design time.

4 The FAERIE AmI Software

FAERIE [4,5] is an AmI framework intended to provide a general infrastructure to develop this kind of system. It has a particular focus on context management, and its design looks to integrate different approaches on this issue. Therefore, it provides a comprehensive starting point to explore the suitability of our integration approach.

FAERIE considers physical spaces where *nodes* work. A *node* is a computational device and the framework components running on it. These components implement the system logics. The devices can be sensors and actuators, but also other computational resources, for instance to provide additional computational power or storage capabilities. An AmI system includes several nodes, possibly in different spaces. These nodes communicate and influence each other through

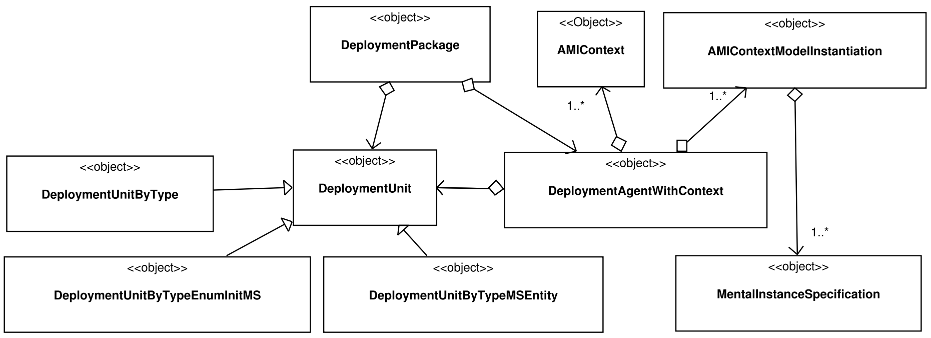


Fig. 4. INGENIAS metamodel fragment of deployment using a context

their physical spaces, e.g. changing the temperature or turning on the lights, and using networks.

Among components, FAERIE distinguishes *context containers* and *context observers*. A *context container* behaves as a blackboard that contains the abstract representation of the context as *context elements*. A set of *context observers* works on these context elements. When there is a change in the representation of the context, the involved *context elements* notify it to the interested (i.e. subscribed) *context observers*, which are able to modify their behavior accordingly.

Context elements can be of three possible types: *Entity*, *Attribute* and *Relationship*. *Attributes* hold information as *ContextValues*. A *ContextValue* stores a data of a primitive type, and information for its management including its type, producer and creation timestamps. *Entities* group all the information (i.e. *Attributes*) related to a concept or object of the domain. *Relationships* represent association between *Entities* according to certain criteria, e.g. the entities that describe the profile of a user or the information needed to carry out an activity.

5 Integrating FAERIE with the INGENIAS Modeling Language

The particularization of the extended INGENIAS meta-model for AmI (see Sect. 3) to FAERIE has two main aspects: the modeling of context information; and the components that manipulate the context.

Figures 5 and 6 show the elements and their relationships used to model context in FAERIE. The required elements (see Fig. 5) are a direct representation of those discussed in Sect. 4. The root of the hierarchy is occupied by *FAERIE Context Element*. This element inherits from *Frame Fact*, which is an INGENIAS entity used to store information within the agent and with the capability of defining *slots* of whatever type. Their relationships are included in Fig. 6. An *agent* can update or receive as notification any *attribute*, *entity* or *relationship*. *Relationships* define constraints and dependencies among *entities*. Attributes are

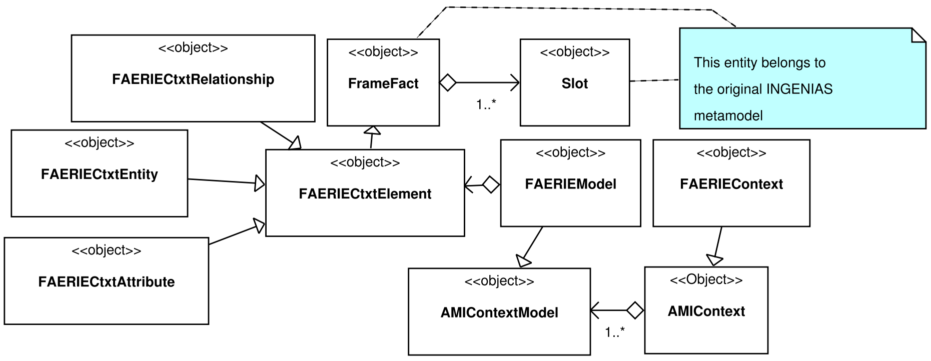

<<object>> FAERIECtxtRelationship
<<object>> FrameFact
<<object>> Slot
1..*
This entity belongs to the original INGENIAS metamodel
<<object>> FAERIECtxtEntity
<<object>> FAERIECtxtElement
<<object>> FAERIEModel
<<object>> FAERIEContext
<<object>> FAERIECtxtAttribute
<<object>> AMIContextModel
<<Object>> AMIContext
1..*

Fig. 5. INGENIAS Meta-model extension to take into account FAERIE entities

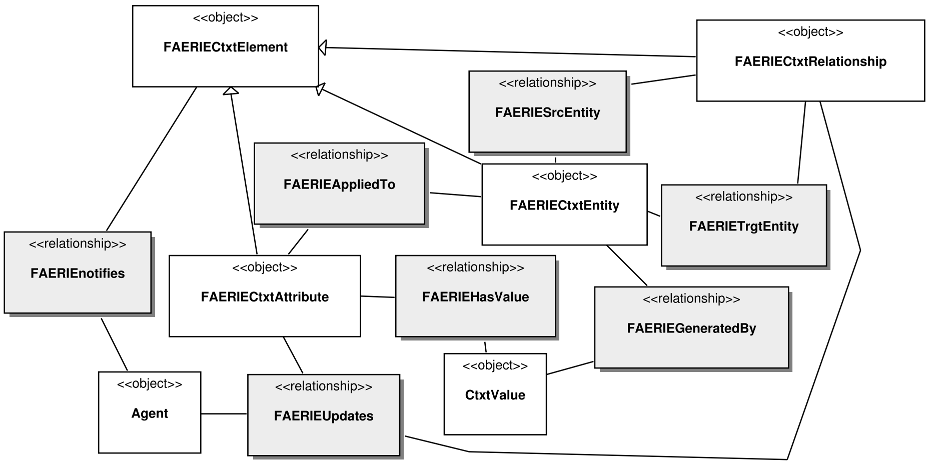

<<object>> FAERIECtxtElement
<<object>> FAERIECtxtRelationship
<<relationship>> FAERIESrcEntity
<<relationship>> FAERIEAppliedTo
<<object>> FAERIECtxtEntity
<<relationship>> FAERIETrgtEntity
<<relationship>> FAERIEnotifies
<<object>> FAERIECtxtAttribute
<<relationship>> FAERIEHasValue
<<relationship>> FAERIEGeneratedBy
<<object>> Agent
<<relationship>> FAERIEUpdates
<<object>> CtxtValue

Fig. 6. Necessary elements to model the FAERIE context

associated to entities by means of *applied to* relationships. *Entities* are expected to produce *values*, and these *values* are associated as well to *attributes*. The *FAERIE value* has been omitted in the meta-model, since it is too low level and does not contribute much to the general design. The kind of values an entity can have can be represented equally thanks to the slot representation capability obtained from *Frame Fact*.

6 Case Study: Tracking Teachers

The previous meta-models have been tested in several case studies to evaluate how well they capture the relevant information to build AmI systems. These meta-models have been fed into the INGENME infrastructure to generate tailored editors that facilitate these studies. This section reports part of a case study for a tracking application related with tutorship in a school. It considers the elements from the FAERIE meta-model introduced in figures 6 and 5.

Students go to tutorships when they need personalized help on some subject. However, teachers are frequently out of their offices and in some other place in the building, for instance in a school board meeting or a conference. The considered AmI application help these students to locate their teachers. It is assumed there are no in-door location facilities, so it is up to the teacher to inform the students where is he. In addition, it is expected the teacher to tell the ambients of the room that he is here. This serves to trigger specific actions, like turning on the heater or the air conditioned, depending on the preferences of the teacher.

The problem is depicted in figure 7. There are four agent types involved in the problem: *Time table context updater* (responsible of keeping track of the teacher time table), *Location context updater* (responsible of deducing current location of the teacher), *Availability context updater* (responsible of observing the teacher location and informing concrete places that the teacher is inside), and Someone application (which represents a third party application that wants to know where the teacher is to trigger concrete actions, like turning on the heater or the air conditioned). The problem involves three context types, two for representing different kind of places, Classroom context and Office context, and one representing the personal context of the teacher. Agents do receive notifications from the contexts and updates them when it makes sense. In this case, the Time context updater will change the time table to indicate which activity the teacher is doing now (like lecturing students in room 5), the *Location context updater* receives notification of the change in the time table and updates the current location of the teacher, the *Availability context updater* is notified of the change in the current location, and proceeds to modify the information within the classroom or office contexts. This later change will trigger additional actions in the third party applications represented by *Someone application*.

Figure 7 served to depict the general scenario. Now, to understand what is modified and what is notified, it is necessary to explore the models associated to the contexts. This is done in figure 8. The models represent dependencies among the information models which each context contains. The *Office* and *Classroom contexts* only require a piece of information, *Teacher availability* telling if the teacher is present. This *availability* is associated to the *teacher* into another model belonging to the *Personal context*. This *context* separates on the one side private information from the user, such as its *identify*, and public information (at least during working hours) of the user, such as *My location* and its *timetable*. A timetable is an attribute of the teacher, like the availability. The location, nevertheless is associated as an attribute to the person. The availability will notify the *Someone application* agent when a change is triggered by the *Availability context updater* agent. The change will start when a modification is notified to this agent by *My location*. This entity will be updated by the *Location context Updater*, which will be notified by a change in the *timetable*. Changes in the *timetable* are oredered by *Timetable context updater*.

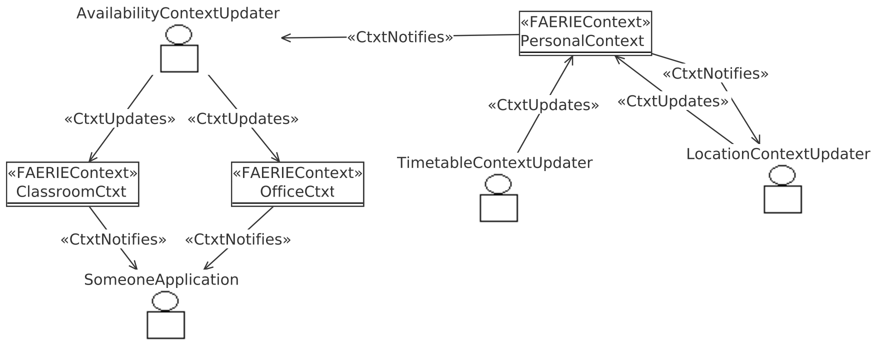


Fig. 7. Agents association with contexts for the teacher case study

Figure 8 can be troublesome to understand. It is a typical case of information overload in a diagram. INGENIAS provides in its new versions with facilities to implode or explode some conflicting entities, like the *context models*. Figure 9 is the same figure as 8 but with all context models collapsed. The readability is improved greatly and permits to focus the attention in concrete contexts each time without having to create separated diagrams.

The deployment of the system requires on the one side identifying the deployed agents, and, on the other side, the contexts to be created. Figure 10 introduces three deployment packages representing a node in one office, another node in one classroom, and a third node representing the user. The user carries one instance of the *Timetable context updater* and the *Location context Updater*. The other contexts have one instance of the *Availability context updater* each one. These instances would be responsible of updated the corresponding instance of the *Teacher Availability* entity, which these contexts will hold. The agent will also request listening to changes in the location through context specific tasks, as figure 12 will point out. There is also two instances of the third party application per node. The instantiation of the associated context models is expressed in figure 11.

The instantiation is expressed in figure 11 as a collection of instances of *Mental entity instances*. These entities do refer to a concrete type (it is the *entity* attribute) and determine the value of slots previously associated to those entities. Here, it is appreciated that instances of the availability entities are associated to the same teacher instance. This is common in this kind of problems and should not be assumed it is exactly the same instance, but an entity with the same information. If two entities share the same information, they are regarded as the same. Nevertheless, this is something to be handled by the FAERIE infrastructure.

To conclude, the agents do have tasks dealing with the events that are produced from the binded contexts. These events will point at changes and will indicate as well old values. The *context* where the event originated is mentioned as well, since the agent may be listening to several contexts. In the figure 12 the task *Register As Observer* is a special *Context Use Task*, which means it has

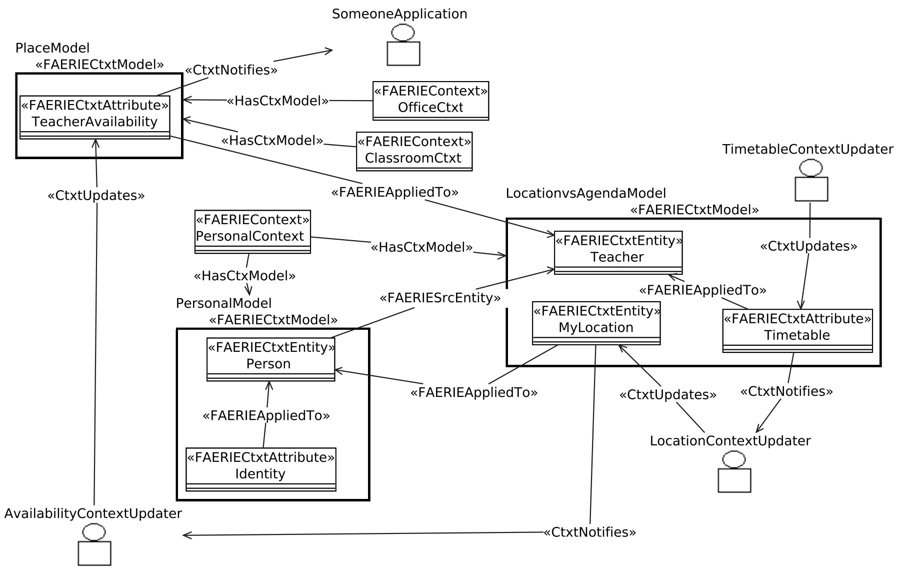


Fig. 8. Context models used in the teacher finding problem

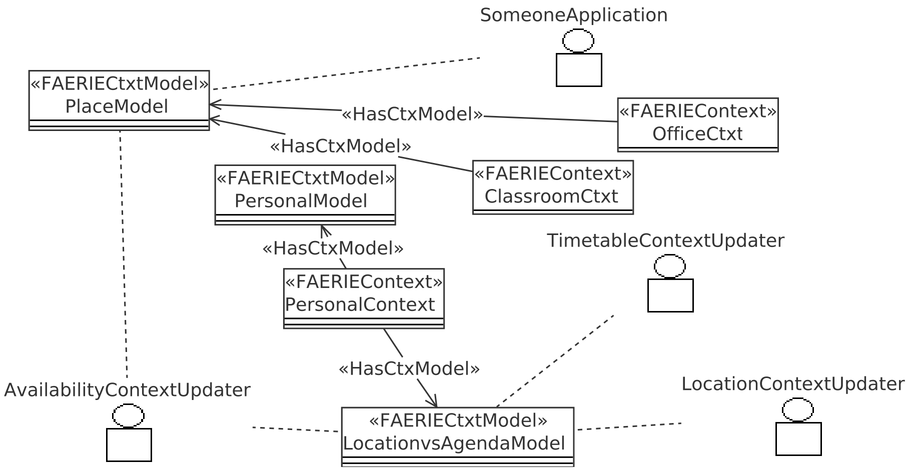

SomeoneApplication
«FAERIECtxtModel» PlaceModel
«HasCtxModel»
«FAERIEContext» OfficeCtxt
«HasCtxModel»
«FAERIECtxtModel» PersonalModel
«FAERIEContext» ClassroomCtxt
«HasCtxModel»
TimetableContextUpdater
«FAERIEContext» PersonalContext
«HasCtxModel»
AvailabilityContextUpdater
LocationContextUpdater
«FAERIECtxtModel» LocationvsAgendaModel

Fig. 9. Simplified version of figure 8 with collapsed context models

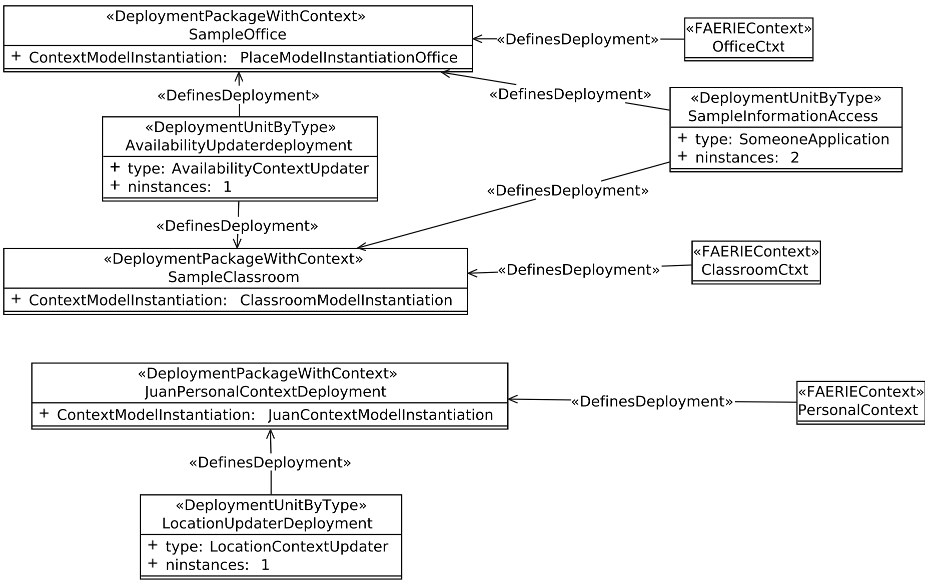

«DeploymentPackageWithContext»
SampleOffice
+ ContextModelInstantiation: PlaceModelInstantiationOffice
«DefinesDeployment»
«FAERIEContext»
OfficeCtxt
«DefinesDeployment»
«DefinesDeployment»
«DeploymentUnitByType»
SampleInformationAccess
+ type: SomeoneApplication
+ ninstances: 2
«DeploymentUnitByType»
AvailabilityUpdaterdeployment
+ type: AvailabilityContextUpdater
+ ninstances: 1
«DefinesDeployment»
«DefinesDeployment»
«DeploymentPackageWithContext»
SampleClassroom
+ ContextModelInstantiation: ClassroomModelInstantiation
«DefinesDeployment»
«FAERIEContext»
ClassroomCtxt
«DeploymentPackageWithContext»
JuanPersonalContextDeployment
+ ContextModelInstantiation: JuanContextModelInstantiation
«DefinesDeployment»
«FAERIEContext»
PersonalContext
«DefinesDeployment»
«DeploymentUnitByType»
LocationUpdaterDeployment
+ type: LocationContextUpdater
+ ninstances: 1

Fig. 10. Deployment of agents and contexts for the teacher finding problem

access to context facilities, like subscribing to changes in an entity in some remote context. Changes are processed by the *Modify Availability In Context* task, which is again a *Context Use Task*. This time, it uses the context update information service to change the values of the *Teacher Availability*. Similarly, the *Someone application* agent has a regular tasks processing events from the context. No special tasks are required in this case since no context specific services are needed.

7 Related Work

The use of visual modeling languages in AmI to address the integration of different approaches is a novelty of this work to the best of our knowledge. For instance, there are no examples in the extensive Sadri's review [12]. However, formal and visual modeling languages have already been applied in this domain.

Formal non-visual modeling appears in some works, particularly when there is need to carry out complex reasoning on context information. For instance, Bosse et al. [3] propose a generic agent framework for monitoring humans when something more sophisticated than evaluating a threshold is necessary. Their models are expressed using a Temporal Trace Language (TTL) for the formal specification and verification of dynamic properties. The framework assumes there are different agents interacting to combine their information and achieve useful conclusions. As in other works, agents are used to wrap ambient sensors and

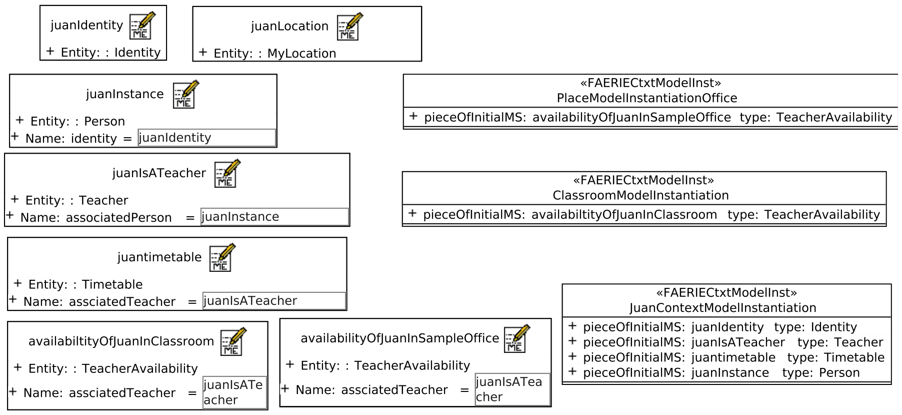

juanIdentity
+ Entity: : Identity
juanLocation
+ Entity: : MyLocation
juanInstance
+ Entity: : Person
+ Name: identity = juanIdentity
juanIsATeacher
+ Entity: : Teacher
+ Name: associatedPerson = juanInstance
juantimetable
+ Entity: : Timetable
+ Name: assciatedTeacher = juanIsATeacher
availabiltityOfJuanInClassroom
+ Entity: : TeacherAvailability
+ Name: assciatedTeacher = juanIsATeacher
availabilityOfJuanInSampleOffice
+ Entity: : TeacherAvailability
+ Name: assciatedTeacher = juanIsATeacher
«FAERIECtxtModelInst» PlaceModelInstantiationOffice
+ pieceOfInitialMS: availabilityOfJuanInSampleOffice type: TeacherAvailability
«FAERIECtxtModelInst» ClassroomModelInstantiation
+ pieceOfInitialMS: availabiltityOfJuanInClassroom type: TeacherAvailability
«FAERIECtxtModelInst» JuanContextModelInstantiation
+ pieceOfInitialMS: juanIdentity type: Identity
+ pieceOfInitialMS: juanIsATeacher type: Teacher
+ pieceOfInitialMS: juantimetable type: Timetable
+ pieceOfInitialMS: juanInstance type: Person

Fig. 11. Instantiation of the context model elements for the teacher finding problem

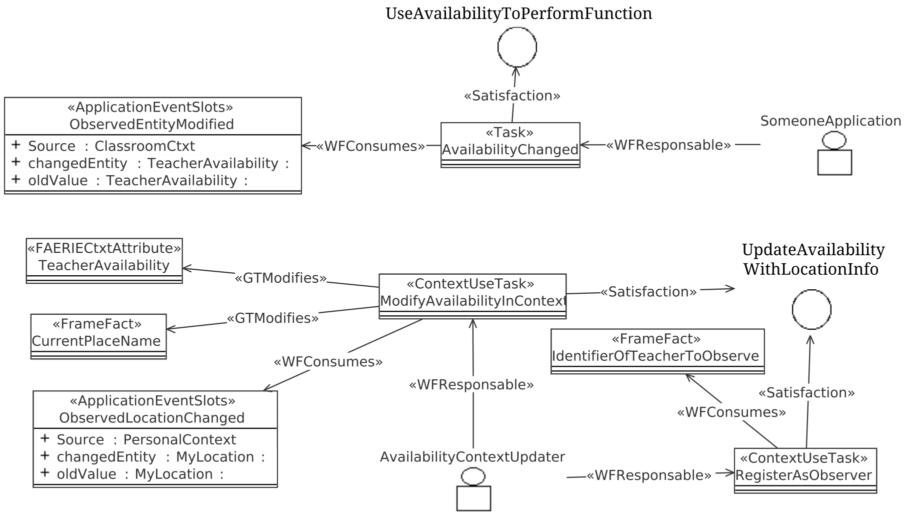

UseAvailabilityToPerformFunction
«Satisfaction»
«ApplicationEventSlots» ObservedEntityModified
+ Source : ClassroomCtxt
+ changedEntity : TeacherAvailability :
+ oldValue : TeacherAvailability :
«WFConsumes»
«Task» AvailabilityChanged
«WFResponsable»
SomeoneApplication
«FAERIECtxtAttribute» TeacherAvailability
«GTModifies»
«ContextUseTask» ModifyAvailabilityInContext
«Satisfaction»
UpdateAvailability WithLocationInfo
«FrameFact» CurrentPlaceName
«GTModifies»
«FrameFact» IdentifierOfTeacherToObserve
«WFConsumes»
«WFResponsable»
«Satisfaction»
«ApplicationEventSlots» ObservedLocationChanged
+ Source : PersonalContext
+ changedEntity : MyLocation :
+ oldValue : MyLocation :
«WFConsumes»
AvailabilityContextUpdater
«WFResponsable»
«ContextUseTask» RegisterAsObserver

Fig. 12. Agents association with context for the teacher case study

actuators. Most of the effort is made at the TTL level, which has an important advantage: it is independent of implementation and permits to ensure some properties are held.

The use of visual modeling languages is quite popular in AmI, as in other realms of Software Engineering. Specifically for context, the GraphicalCM [7] is a visual modeling language that allows mapping abstract models described with GraphicalCM to implementation models. It is interesting how associations between elements are labeled with quality parameters. GraphicalCM does not consider the agent concept, but it can be used with our approach to enrich the modeling language used for the context.

8 Conclusions

This paper has introduced extensions of the INGENIAS modeling language to include AmI aspects. In this first try, the focused on the modeling language to integrate properly with INGENIAS and to address concrete problems identified in a FAERIE development. As a result, several extensions to the INGENIAS meta-model have been regarded. These extensions are part of the INGENIAS editor v1.4 which can be used from the Maven repository or following the instructions from `http://ingenias.sf.net`. These extensions are intended to be general in the future and need to address other platforms. The first step has used the FAERIE platform, an ongoing project that already integrates several works on context-aware applications.

The use of these extensions has been exemplified in this paper with a case study on a tracking application. It uses different sources of information (e.g. timetable and location) to locate a teacher within a building and determine if s/he is available for tutorship. These extensions are not completed yet. There are important areas that require work, specially privacy and the dynamic of the context. Privacy is a critic concern in an AmI development which intends to be open as ours. We have not addressed this problem yet and it may involve dealing as well with the incorporation of other agents. FAERIE does consider privacy and authority, but we believe those concerns can be considered in an increment of the language without modifying existing entities. Another related issue is context dynamics, concretely, the subscription to a context. The cycle depicted in figure 2 does not regard the possibility of being rejected or the need to meet some conditions in order to be accepted. Also, the behavior of the agents within the context ought to be a concern as well. Agents abusing the context services ought to be expelled. Research in norms, regulations, and institutions may be very relevant here. To conclude, the modeling language is not informing to which contexts an agent can have access. In fact, we considered to enrich the agent declaration indicating which context models are known. However, this conflicts the simplicity of the context notifies/updates approach, which, implicitly, tells the agent must know the context models used within the updated/notifier context. This decision will have to be reviewed after some additional experimentation.

Acknowledgements. We acknowledge support from the project "SOCIAL AMBIENT ASSISTING LIVING - METHODS (SociAAL)", supported by Spanish Ministry for Economy and Competitiveness, with grant TIN2011-28335-C02-01 and by the Programa de Creación y Consolidación de Grupos de Investigación UCM-Banco Santander for the group number 921354 (GRASIA group)

References

1. Aarts, E., Wichert, R.: Ambient intelligence. In: Bullinger, H.-J. (ed.) Technology Guide, pp. 244–249. Springer, Heidelberg (2009)
2. Bolchini, C., Curino, C.A., Quintarelli, E., Schreiber, F.A., Tanca, L.: A data-oriented survey of context models. SIGMOD Rec. 36(4), 19–26 (2007)

3. Bosse, T., Hoogendoorn, M., Klein, M.C.A., Treur, J.: An ambient agent model for monitoring and analysing dynamics of complex human behaviour. JAISE 3(4), 283–303 (2011)
4. Fernández de Alba, J.M., Pavón, J.: Recognition and interpretation on talking agents. In: García-Pedrajas, N., Herrera, F., Fyfe, C., Benítez, J.M., Ali, M. (eds.) IEA/AIE 2010, Part I. LNCS, vol. 6096, pp. 448–457. Springer, Heidelberg (2010)
5. Fernández de Alba, J.M., Pavón, J.: Talking agents in ambient-assisted living. In: Setchi, R., Jordanov, I., Howlett, R.J., Jain, L.C. (eds.) KES 2010, Part IV. LNCS, vol. 6279, pp. 328–336. Springer, Heidelberg (2010)
6. German Commssion for Electrical, Electronic and Information Technologies of DIN and VDE. The German AAL Standardization Roadmap. Technical report, VDE Association for Electrical, Electronic and Information Technologies (2012)
7. Henricksen, K., Indulska, J., Rakotonirainy, A.: Modeling context information in pervasive computing systems. In: Mattern, F., Naghshineh, M. (eds.) PERVASIVE 2002. LNCS, vol. 2414, pp. 167–180. Springer, Heidelberg (2002)
8. Pavón, J., Gomez-Sanz, J.J., Fuentes-Fernández, R.: Understanding agent oriented software engineering methodologies. In: Agent-Oriented Software Engineering X - 10th International Workshop, AOSE 2011, Taipei, Taiwan (2011)
9. Kelly, S., Lyytinen, K., Rossi, M.: Metaedit+: A fully configurable multi-user and multi-tool case and came environment. In: Constantopoulos, P., Mylopoulos, J., Vassiliou, Y. (eds.) CAiSE 1996. LNCS, vol. 1080, pp. 1–21. Springer, Heidelberg (1996)
10. Newell, A.: The knowledge level. Artificial Intelligence 18(1), 87–127 (1982)
11. Pavón, J., Gómez-Sanz, J.J.: Agent oriented software engineering with INGENIAS. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS (LNAI), vol. 2691, pp. 394–403. Springer, Heidelberg (2003)
12. Sadri, F.: Ambient intelligence: A survey. ACM Computing Surveys (CSUR) 43(4), 36 (2011)

Analysing the Suitability of Multiagent Methodologies for e-Health Systems

Emilia Garcia[1], Gareth Tyson[2], Simon Miles[3], Michael Luck[3], Adel Taweel[3], Tjeerd Van Staa[4], and Brendan Delaney[3]

[1] Universitat Politecnica de Valencia, Spain
[2] Queen Mary, University of London
[3] King's College London, UK
[4] General Practice Research Database, UK

mgarcia@disc.upv.es, gareth.tyson@eecs.qmul.ac.uk,
{simon.miles,michael.luck,adel.taweel,brendan.delaney}@kcl.ac.uk,
tjeerd.vanstaa@gprd.com

Abstract. Online e-health systems are being proposed and developed at an ever increasing rate. However, the progress relies on the interoperability of local healthcare software, and is often hampered by ad hoc methods leading to closed systems with a multitude of protocols, terminologies, and design approaches. Agent-oriented software engineering (AOSE) seems intuitively a good approach for developing more open systems. While agent-based e-health systems have been developed, the general hypothesis of the suitability of AOSE has not been evaluated. In this paper, we test that hypothesis, including a case study of applying a normative agent methodology to a particular real-world e-health system, and present an analysis of the strengths and weaknesses of AOSE for e-health.

Keywords: Health systems, Normative environments, Organisational Agent Architectures, Contracts, System of Systems.

1 Introduction

Large-scale and flexible systems are increasingly needed to fulfil the emerging requirements of complex domains. One typical example is *healthcare*, which is rapidly becoming more and more dependent on large-scale integrated software systems. On the one hand, these systems offer new and innovative ways to improve patient care; however, on the other, they also introduce complications and risks that were never envisaged in the early days of healthcare computerisation. Clearly, these complications affect the development of related software. A particular challenge is that the healthcare domain is separated into many disparate organisations that often fall under different spheres of control. As a result, it is common for systems to be constructed out of many divergent sub-systems; this is termed a *system of systems* (SoS). In this context, interactions can often take place between components that are managed by parties with conflicting goals, different policies, incompatible data representations, and so on. Not surprisingly, this can lead to serious challenges when integrating these different systems in a trustworthy, consistent manner, leading to the emergence of strict regulatory controls to manage not only the internal behaviour of organisations, but also the interactions that may take place between multiple organisations.

While multi-agent technology has emerged over the last decade as a new software engineering paradigm for building complex, adaptive systems in distributed, heterogeneous environments, it is still not mainstream in its domain application. Nevertheless, it can be observed that many properties of the healthcare domain fit well with several concepts that arise in the area of multi-agent systems, such as organisational autonomy, inherent regulatory frameworks, and so on. It thus seems appropriate that introducing such principles to the development of healthcare systems could offer many benefits. Indeed, there have been several agent-based e-health systems developed over a period of many years [3,37], but they rarely employ explicit agent-oriented software engineering (AOSE) methodologies and, as such, do not directly evaluate the suitability of AOSE to this domain in particular.

Addressing this omission, in this paper we investigate the suitability of using AOSE, and the common underlying concepts used in AOSE design and development, for the creation of e-health systems. We wish to answer the following question: *To what extent is AOSE an approach that is appropriate for the development of e-health systems?*

To represent AOSE in testing this hypothesis, we take a specific methodology, ROMAS (Regulated Open Multi-agent Systems). ROMAS is an AOSE methodology that guides developers all the way from the requirements analysis phase to the actual implementation, taking into account the notions of agents, organisations, services and contracts. As ROMAS shares many of the same fundamental concepts with other existing AOSE methodologies, it can be seen as an adequate representative for testing the hypothesis. In this paper we apply the ROMAS methodology to a particular (real) e-health system: ePCRN-IDEA [47,46]. This system allows us to exemplify the features of healthcare systems, so as to evaluate the suitability of AOSE in addressing them.

The rest of the paper is organised as follows. Section 2 details the ePCRN-IDEA case study and summarises the main challenges of the development of e-health systems. Section 3 introduces some common AOSE design abstractions that seem useful in the analysis and design of e-health systems. Section 3.2 introduces the ROMAS methodology. Section 4 summarises our proposal for developing the case study using ROMAS and analyses the benefits that the ePCRN-IDEA system has obtained by means of using AOSE techniques. Section 5 analyses the suitability and general benefits of using agent methodologies in the analysis and design of e-health systems. We identify a number of strengths and weaknesses of AOSE for such systems, as well as suggesting improvements to better support the needs of the domain. Finally, Section 6 summarises the results of this work.

2 Case-Study: ePCRN-IDEA System

In this section we present a system architecture for recruiting patients for clinical trials in real-time. First, Section 2.1 introduces the real-life case study and its domain context. Second, Section 2.2 discusses the challenges of designing an e-health system using the case study as example.

2.1 ePCRN-IDEA Overview

Clinical trials are experiments by which the efficacy of medical treatments are explored. They involve recruiting patients with specific characteristics to undergo new treatments, so that the effectiveness and safety of those treatments can be tested. However, a key challenge in this is recruiting sufficient patients to ensure that the results are meaningful. This has long been a difficult problem as the requirements for participation are often very strict, making it difficult to locate eligible patients. ePCRN-IDEA is a new system under deployment in the UK healthcare system that notifies practitioners in real-time whenever an eligible patient is in consultation. When a patient visits a clinic, ePCRN-IDEA compares their details against a database of trials; if the patient is eligible for one or more, the practitioner is prompted to try to immediately recruit the patient if they are interested. Further details of the ePCRN-IDEA project can be found in the presentation by Tyson et al. [46].

2.2 Challenges in ePCRN-IDEA's Development

Development of the ePCRN-IDEA system [47] has identified a number of core challenges, which are typical of similar systems in the health domain. In this light, this section briefly covers the most important of these identified challenges to gain a better understanding of how AOSE might be able to benefit the development process of such systems.

Integration of Independent Systems. In order to recruit eligible patients, it is necessary for researchers, practitioners, patients, databases and clinics to interact. This means that several independent institutions, which are completely autonomous and have their own independent goals, must cooperate to achieve a common objective. However, the integration of multiple heterogeneous and autonomous systems can be a complicated and resource-consuming task. Some of the issues that must be solved are [42,41]:

– *Distributed Data*: the required data is spread widely across all organisations, frequently using different schemas;
– *Technical Interoperability*: different organisations often use different (potentially incompatible) technologies;
– *Process Interoperability*: different organisations often employ divergent (potentially incompatible) processes to achieve their goals;

- *Semantic Interoperability*: different organisations often utilise different vocabularies and coding schemes, making it difficult to understand the data of others;
- *Trustworthiness*: little trust exists between different organisations, particularly those with conflicting goals and interests. In consequence, healthcare systems that consist of multiple organisations must take all these aspects into account to ensure successful operation.

Regulation of Independent Systems. Healthcare systems must fulfil strict governmental regulations concerning the privacy and security of personal patient data. Moreover, each research institute and clinic has its own regulations, specific goals, priorities and restrictions to regulate the behaviour of each of its members. Healthcare systems must therefore often take into account several regulatory environments.

System Evolution. Medical institutions are constantly adapting their systems to reflect new legislation, software and medical techniques. As these autonomous organisations often operate with a range of aims and priorities, it is possible that changes may take place without necessarily propagating to all other parts of the system. In this respect, a change within one sub-system could result in violations of responsibilities in another sub-system (e.g. by changing data formats). Healthcare systems that consist of multiple organisations must therefore ensure some formal procedure by which all parties understand and adhere to their responsibilities. To enable practical deployment, institutions must also be contractually obliged to adhere to a standard interaction mechanism and data format, although their internal process or storage technology changes.

3 Agent-Oriented Software Engineering

Multi-agent systems (MAS) are used in real-industrial applications for developing complex, adaptive systems in distributed, heterogeneous environments in different domains. Although there is much current interest in the use of MAS approaches for the development of healthcare systems, the MAS paradigm is still not mainstream in this domain. Several agent-based e-health systems have been developed [27,39], but the use of agent-oriented software engineering (AOSE) methodologies to analyse and design these kinds of systems has not extensive.

In this section we introduce some common AOSE design abstractions that seem useful in the analysis and design of e-health systems. We also introduce the ROMAS methodology in order to analyse and design the case study. Further details of the ROMAS methodology can be found in [23,24].

The ROMAS methodology is considered an adequate representative for testing the hypothesis of this paper because it integrates in its development process all the AOSE design abstractions detailed in Section 3.1. It also offers a set of guidelines that facilitate the analysis and design process, such as guidelines for selecting the most suitable social structure, and for identifying and formalising the normative context of a system. However, it is important to highlight that the conclusions of this paper are not related to the specific features of ROMAS, but instead to general features of AOSE methodologies.

3.1 AOSE Design Abstractions

In MAS, agents are entities characterised by their autonomy, reactivity, proactivity, and social ability.

The concept of *organisation* has become a key concept in MAS research, since its properties can provide significant advantages when developing agent-based software, allowing more complex system designs to be built with a reduced set of simple abstractions [30,31]. Organisations comprise both the integration of organisational and individual perspectives and the dynamic adaptation of models to organisational and environmental changes. Relevant organisational methodologies are: Gaia [49], AML [45], AGR [19], AGRE [20], MOISE [25], INGENIAS [38], OperA [14], OMACS [10]. A detailed survey of organisational approaches to agent systems can be found in [48].

Another concept that it is gaining importance in MAS is the integration of service-oriented architectures into MAS [22]. A service-oriented open MAS (SOMAS) is a multi-agent system in which the computing model is based on well-defined, open, loosely-coupled service interfaces such as *web services*. Such services can support several applications including: heterogeneous information management; scientific computing with large, dynamically reconfigurable resources; mobile computing; pervasive computing; etc. Relevant SOMAS proposals are: Tropos [8], Alive [13], GORMAS [1], INGENIAS [21].

Agents that join an organisation usually have to deal with some constraints, such as the need to play particular roles so as to participate in certain allowed interactions. Other higher-level abstractions are normally employed, such as *norms* for keeping agents from unexpected or undesirable behaviour [29]. Currently, the most developed agent methodologies integrate norms into their meta-models in order to formalise restrictions on the behaviour of the actors of the systems [4,11,16,2]. Many of them also allow the specification of organisational systems. These agent methodologies are able to describe different normative contexts by means of specifying norms whose scope is limited to one specific organisation of the system [43,15,7].

Over the last few years, the integration of electronic *contracts* in MAS is becoming increasingly more important to system architectures for agent behaviour regulation [40,34,32]. Most approaches integrate contracts in order to specify the contractual agreements between parties [28,7], but few approaches use contracts to specify the structure of the system and the social relationship among the system's entities [6,36,15].

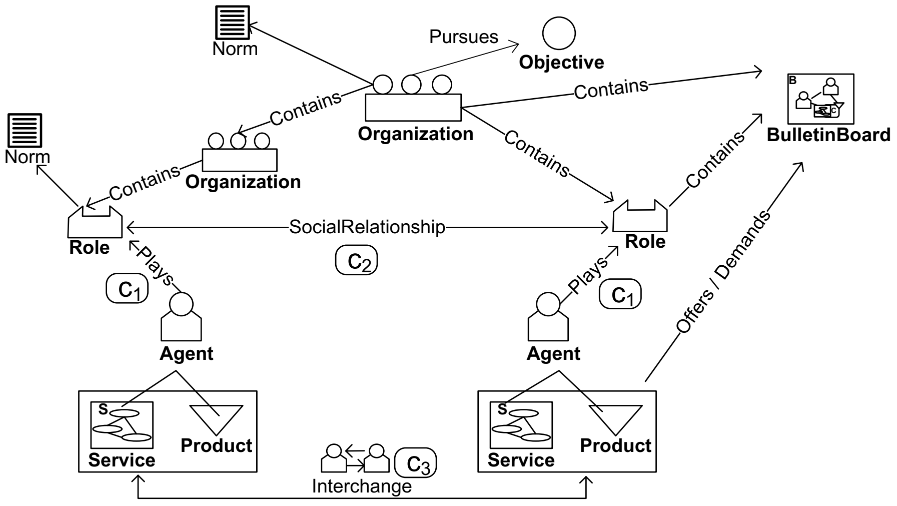


Fig. 1. Overview of ROMAS architecture

3.2 ROMAS (Regulated Open Multi-agent Systems)

In this section we introduce how ROMAS integrates the common AOSE concepts of agents, roles, organisations, norms and contracts. A complete description of ROMAS can be found in [23,24], so we address only the key aspects here. In ROMAS, *agents*, *roles* and *organisations* are defined through a formal social structure based on a service-oriented open MAS architecture, whose main features are summarised in Figure 1. The graphical notation used in ROMAS models is based on the notation used in other methodologies like Ingenias [38] and Anemona [26].

Organisations in ROMAS represent a set of individuals and institutions that need to coordinate resources and services across institutional boundaries. In this context, agents represent individual parties who take on roles in the system; within a given organisation (e.g. a company), they can both offer and consume services as part of the roles they play. Beyond this, virtual organisations can also be built to coordinate resources and services across institutional boundaries. Importantly, each of these concepts must be strictly defined, alongside their interrelations. Organisations are conceived as an effective mechanism for imposing not only structural restrictions on their relationships, but also normative restrictions on their behaviour. These restrictions are formalised in ROMAS by means of norms and contracts.

Norms in ROMAS are specified using the model described in [9] which defines norms that control agent behaviour, the formation of groups of agents, the global goals pursued by these groups and the relationships between entities and their environment. Specifically, it allows norms to be defined: (i) at different social levels (e.g. interaction and institutional levels); (ii) with different norm types (e.g. constitutive, regulative and procedural); (iii) in a structured manner; and (iv) dynamically, including later derogation. Figure 1 shows two types of norms: (i) those that are associated with each organisation; and (ii) those that are associated with each role. Clearly, the former must be complied with by any organisation member, while the latter must be complied with by all agents playing that role.

Finally, ROMAS also allows interactions to be formalised by means of *contracts*. These are necessary when working in an open regulated system, to be able to specify the expected behaviour of others without compromising their specific implementation. ROMAS involves two types of contracts: *social contracts* and *contractual agreements*. Social contracts can be defined as statements of intent that regulate behaviour among organisations and individuals. As shown in Figure 1, such social contracts are used to formalise relationships: (i) between an agent playing a role and its host organisation (as indicated by the contract labelled c_1); and (ii) between two agents providing and consuming services (as indicated by c_2). Social order thus emerges from the negotiation of contracts over the rights and duties of participants, rather than being specified in advance. In contrast, contractual agreements represent the commitments between several entities in order to formalise an interchange of services or products (c_3).

4 Analysis of the Case Study

This section describes the design of the ePCRN-IDEA recruitment system using the ROMAS methodology. It also analyses how an agent methodology can deal with the main challenges of the development of this system.

4.1 Designing ePCRN-IDEA Recruitment System with ROMAS

In this section, we present the ePCRN-IDEA system design following the ROMAS methodology. Figure 2 shows the main structure of ePCRN-IDEA in terms of the key concepts of organisations, roles, norms and contracts, detailed below.

Organisations and Processes. Several organisations are involved in the key processes performed in ePCRN-IDEA, as follows. When a research body wishes to create a new clinical trial, they can inject it through a service called the Central Control Service (CCS), which is hosted at *King's College London* (KCL). The CCS stores trials within a large database in a pre-defined format that all researchers must adhere to. Associated with each trial is a list of potentially eligible patients; these lists are generated by the *General Practice Research Database* (GPRD), which operates a large data warehouse containing over 12 million up-to-date patient records in the UK. Following this, the trials and their eligibility lists are distributed to software agents (called LEPIS agents) that operate on clinicians' PCs at each participating *clinic*. LEPIS agents then listen to the interactions between the practitioner and their local Electronic Health Record

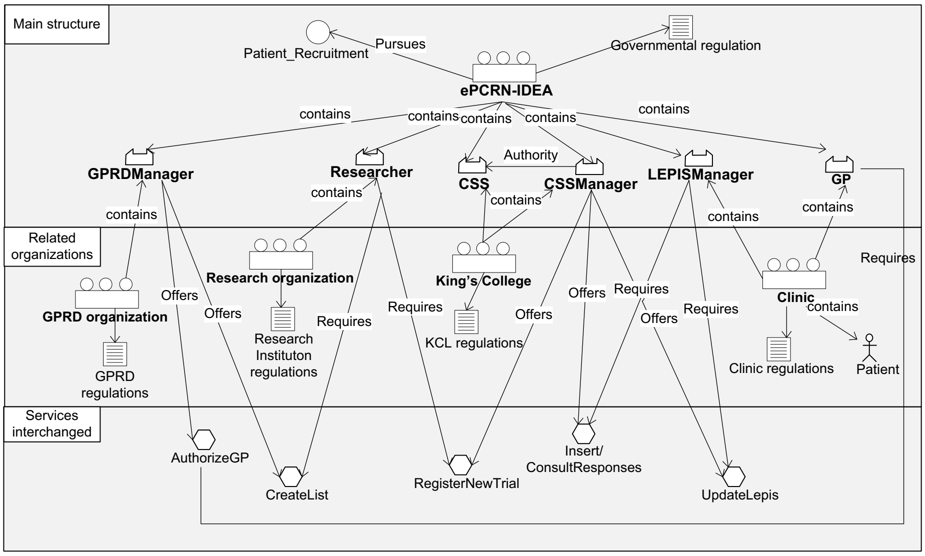


Fig. 2. ePCRN-IDEA organisational structure

(EHR) database, which is used to store information about patients (e.g. diagnoses, treatments, demographic data, etc.). During consultations, LEPIS agents compare the patient information against the eligibility lists of all known trials. If a patient is found to be eligible for a trial, the practitioner is notified, and if the patient is interested, the system loads a Random Clinical Trial (RCT) website provided by the *research body* responsible for the trial, allowing the patient's recruitment to be completed. Consequently, the following organisations are involved: KCL, GPRD, the clinics and the research bodies.

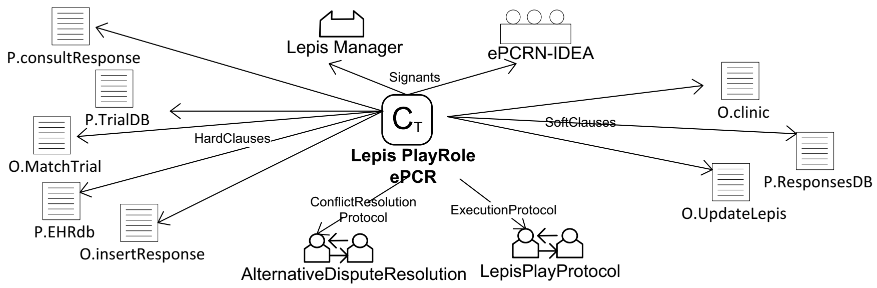
Roles. The system is composed of six different roles presented below.

– The *GPRD Manager Role* is responsible for updating and controlling access to the GPRD database. It offers a service to pre-compute lists of eligible patients for individual trials based on complex search criteria (*CreateList* service). The role must also offer a service to decide when a GP is authorised to perform recruitment for each trial (*AuthoriseGP* service). The agent that plays the GPRD Manager role must also play a role in the governmental body (represented as the *GPRD organisation*), so it must follow the special governmental legislation related to the management of this kind of data.
– The *Researcher Role* is responsible for defining the specific features of each trial under its jurisdiction. Researchers are also responsible for inserting these trials into the CCS database by means of the service offered by the CCS role (described below). They are not allowed to directly contact patients unless they have agreed to participate in a clinical trial under their supervision.

For obvious reasons, each researcher should be part of a specific research institution and follow its specific normative restrictions.

- The *CCS Role* is a software application responsible for controlling the CCS database, which stores data about active clinical trials. It offers three services to the other members of the system: (i) a *Register New Trial* service that allows researchers to inject new clinical trials into the database; whenever a Researcher tries to inject a new trial into the CSS database, the CSS role must verify that this trial follows the specified standards and regulations; (ii) an *Update LEPIS Database* service that allows the clinic's local database to update its information about the active clinical trials; and (iii) an *Insert/Consult Patients Response* service that allows the response of each patient to be registered (whether they agree or refuse to participate in a trial). The current implementation of the CCS role is performed by an agent that is part of the KCL organisation. Clearly, this agent must comply with established norms concerning replication of information, privacy and programmed machine maintenance.
- The *CCS Manager Role* is responsible for controlling the information in the CCS (i.e. it has control over the CCS Role). Due to the specific requirements described by the domain expert, there must be a human responsible for this. This role must be played by a member of KCL, who must therefore comply with the restrictions and rules that KCL establishes.
- The *LEPIS Manager Role* is played by a software application that resides at a clinic and investigates the eligibility of any present patient. There is thus a LEPIS agent playing this role for each clinic participating in the recruitment system. LEPIS agents use the CCS service to acquire information about the clinical trials related to the type of patients for which its clinic is specialised. LEPIS agents also provide the GP with a simple interface to notify them of a patient's eligibility, as well as the option to launch the RCT website if the patient is interested.
- The *GP Role* represents a practitioner working in a clinic. If a GP wants to recruit patients for trials, they must be previously authorised by the GPRD Manager. This authorisation involves the acceptance of some norms related to privacy, and specific restrictions described for each clinical trial. Clearly, each GP must also comply with the rules of their own clinic. Finally, patients are considered external entities for the ePCRN-IDEA system because their interaction with the system is always executed through their GP.

Norms and Contracts. The *Governmental regulations* related to the privacy of patient data and clinical trials are described at a system-wide level; i.e., every agent playing a role inside ePCRN-IDEA should comply with them. At the same time, each institution and clinic defines its own regulations, so the entities of the system should follow the general governmental regulations and the restrictions established by the institution to which they pertain. For instance, each LEPIS agent should follow both global and clinic-specific regulations. The rights and duties that any specific agent implementation must fulfil to play a role in ePCRN-IDEA are formalised by means of a *Social Contract*. Even though contracts



NORM ID	NORM DESCRIPTION (Deontic,Target,Activation,Expiration,Action,Sanction,Reward)
O.MatchTrial	(OBLIGED, Lepis, Event(changesEHR), - , Match_Trial_Historical,-,-)
O.UpdateLepis	(OBLIGED, Lepis,DAILY, - , Request(UpdateLepis service),-,-)
P.EHRdb	(PERMITTED, Lepis, -,-, Read(EHR database),-,-)
P.TrialDB	(PERMITTED, Lepis, -,-, Read(Lepis trial database),-,-)
P.ResponsesDB	(PERMITTED, Lepis, -,-, Write(Lepis patient responses database),-,-)
P.consultResponse	(PERMITTED, Lepis, -,-, Request(ConsultPatientAnswer service),-,-)
O.insertResponse	(OBLIGED, Lepis, GPInsertResponse, - ,Request(InsertPatientResponse service),-,-)
O.clinic	(OBLIGED, Lepis, -,-, Pertain(Clinic),-,-)

Fig. 3. Phase 2: Lepis PlayRole social contract template

are dynamic entities that cannot be completely specified at the design stage, designers can specify the predefined restrictions that all final contracts of a specific type should follow. These restrictions are defined in a *Contract Template*, where *Hard Clauses* indicates mandatory clauses that any contract of this type must contain and *Soft Clauses* indicate more flexible recommendations. Clearly, due to space constraints, a comprehensive set of norms and contracts in ePCRN-IDEA cannot be listed; thus, we briefly cover a small number of examples.

Figure 3 describes the *LEPIS PlayRole* contract template. It specifies that any agent playing the LEPIS Manager role must detect changes in the EHR database and after that it must check the suitability of this patient for any trials (*Norm O.MatchTrial*). The contract template also recommends that the final contract includes a norm specifying that the local LEPIS database must be updated with new clinical trials every day (*Norm O.UpdateLepis*). This clause is merely a recommendation so that at runtime, LEPIS agents are able to negotiate with the ePCRN-IDEA organisation exactly how often they should update their local database. The remaining clauses relate to the use of the local LEPIS databases and the service dependencies that LEPIS requires. In this way, each clinic can implement its own LEPIS agent (if it complies with the required contracts and norms), allowing each clinic to adapt the behaviour of LEPIS in line with its own priorities. For example, a clinic could decide that its LEPIS agent should not increase patient queues; e.g. GPs should not be notified during busy periods. Similarly, each entity that plays any role in ePCRN-IDEA can be adapted to the different requirements and restrictions of its own institution. Each institution would thus maintain its own technology, with different implementations of each role interacting independently of the technological differences.

4.2 Benefits of Multi-agent Design for ePCRN-IDEA

In this section, we revisit the design challenges listed in Section 2.2 to see how effective the use of an agent methodology has been.

Integration of Independent Systems. AOSE offers an effective design platform for modelling and integrating the different ePCRN-IDEA systems by enforcing a high level of abstraction, using many real-world concepts (e.g. organisations). First, this helps domain experts, who are typically not familiar with the relevant technology, to gain a better understanding of the system. Beyond this, it also provides well defined boundaries between different agents and organisations, allowing individual objectives and regulations to be specified, as well as maintaining the privacy of each institution's data and processes. Importantly, technical and semantic interoperability challenges are also addressed by means of standardised web service interfaces.

Regulation of Independent Systems. The regulatory needs of ePCRN-IDEA fit well into the AOSE principles. Specifically, as is shown in the ROMAS design, AOSE techniques allow different normative environments for each clinic and research institution to be explicitly described and combined with global governmental norms. This allows the behaviour of the different entities to be formally constrained — an extremely important feature in the medical domain. Furthermore, different vendors and technologies can be used to implement the agents that play each role. For instance, each clinic could specify and implement its own LEPIS agents according to its individual aims, restrictions and priorities, while maintaining the stability of the system through global governmental regulations. This is particularly important when potentially deploying agents across multiple research institutions and clinics from different countries.

System Evolution. AOSE offers an effective paradigm for assisting in system evolution in ePCRN-IDEA. Through norm and contract regulation, each sub-system can evolve while ensuring that it does not compromise its responsibilities to other parties. Common examples include adaptation to new internal regulations or to the use of a new software technology. Moreover, global system evolution can also take place by publishing new contracts and norms, thereby forcing sub-systems to adapt.

5 Discussion

In the previous sections, we have considered how an existing e-health system could have been developed using an existing agent methodology, and the benefits of doing so. We now consider the more general hypothesis presented in Section 1: that AOSE is highly appropriate for the development of e-health systems. For the evaluation, we can draw not only on our ePCRN-IDEA example, but also on other multi-agent systems in e-health. In general, AI technology including agent-based systems have been used in healthcare to tackle endemic issues such as distributed information and expertise, unpredictable dynamics, uncertainty in reasoning and simulation of systems [39,50].

5.1 Beneficial Features of AOSE

AOSE methodologies commonly include analysis and design based on a few key ideas: *agents* as autonomous, pro-active, flexible and social entities; *interactions* of a flexible and well-defined nature between those agents; and *organisations* in which agents operate, modelled either implicitly or explicitly [12,5,33]. The functionality that agents enact in such designs is sometimes modelled as *services* [22]. Other features present in some methodologies, including ROMAS, are the assumptions of *openness* in the system, and of *regulation* to be followed by agents (e.g. norms, responsibilities, rights, contracts, etc. [11,15,44]). Through the lessons learned during the development of ePCRN-IDEA, we now present some features of AOSE that indicate its suitability for general e-health applications.

Assumption of Autonomy. A critical aspect of e-health systems is that they are comprised of sub-systems that have their own regulations, privacy issues, localised authority, localised flexibility, and so on. For instance, in ePCRN-IDEA, different policies are applied in different clinics and regions in the UK. In this context, it is clear that e-health systems must also take into account this diversity. This stems from factors such as the need to preserve patient confidentiality, the commercial sensitivity of drug development, and from government involvement and regulation at a local level. The autonomy of agents and organisations assumed at the analysis stages in AOSE means that this is a particularly well-suited approach.

Allowance for Openness. There are tens of thousands of independent sites involved in healthcare in various capacities worldwide (with varying levels of system computerisation). A common feature of large-scale e-health systems, such as ePCRN-IDEA, is the expectation that more sites will join the system as they develop the technical capability to do so (e.g. new clinics, research institutes etc.). This means that methodologies with an assumption of an open system are well tailored to e-health. In practice, openness is enabled by a design specifying exactly how a new party must behave in order to join the system, such as through contracts (as in ROMAS) or roles, as well as lower level concerns such as interfaces and interaction protocols.

Explicit Norms. Due to the confidentiality issues mentioned above, healthcare is highly regulated at all levels, and these regulations must be considered as a primary influence on any e-health system. Regulations apply both to individual clinics and researchers, and across the whole system due to national or international laws. For instance, in ePCRN-IDEA each clinic and practitioner must be individually authorised to recruit for each trial. Clearly, e-health also includes norms of good practice that are not strict regulations but with which it is preferable to comply. The advantage of a norm-based design approach is that there is a ready way for developers to specify these regulations explicitly in the development process, such that they become part of the design. Implementing the system in a norm-aware platform can ensure their fulfilment, even if the system has been externally implemented by different providers. For instance, if the system deals with critical restrictions, a *regimented* agent platform like [18] should be used. On the other hand, if the domain of application allows the violation of norms, an enforcement architecture like [9] should be used.

Domain-Like Concepts. Agents, norms and organisations directly map to the important features of the healthcare environment at a high level, for the reasons described above. That is, healthcare specifically concerns people (patients and clinicians), the organisations they work for, and the regulations they must comply with. This aids discussion with domain experts, thereby easing such things as requirements elicitation and verification (though there are limits, as discussed below).

5.2 Other Development Approaches

In theory, an e-health system such as ePCRN-IDEA could be designed by a single organisation in a centralised manner, following one of the many methodologies tailored to single, non-distributed systems. However, as described above, the required data and functionality is distributed among clinics, and their autonomy makes this unrealistic. Also, for most applications, the number of patients, clinics, trials, etc., could produce a scalability problem.

Turning to more comparable development views, a service-oriented approach to development is clearly appropriate in some respects [35,17]. It assumes some autonomy, in that services can be separately hosted and maintained, and allows for some openness, as existing published interfaces may be implemented by new services. However, traditional service-oriented applications are generally controlled ultimately by a single client and the interfaces only partially specify how a service should behave. Service-oriented architectures do not explicitly represent the social structure of the system and the institutions involved in the system. Besides, service-oriented applications do not have explicit social norms (though service-level agreements can act as contracts for low-level quality of service demands).

Methodologies based on concepts of objects or components, regardless of the particularities of the methods themselves, also suffer from having less domain-like concepts than AOSE. This point is not healthcare-specific, but significant in any domain in which the requirements relate to user interaction rather than merely system component interaction. Objects and components will normally have parallels in the domain, but these will be of less direct concern than the people, organisations and regulations.

The comparison above is not to say that services or objects are irrelevant to developing e-health systems, but are inadequate in themselves compared to an AOSE approach. Many AOSE methodologies, including ROMAS, utilise service-oriented and object-oriented specifications of the functionality performed by agents.

5.3 AOSE Weaknesses

There are two weaknesses of the current approach in applying ROMAS to ePCRN-IDEA. Although they are weaknesses of ROMAS, we believe them to apply to current AOSE methodologies more generally.

First, while conceptualising the system in terms of agents, organisations and norms was found to be intuitive by domain experts, the language itself was not. There are terms in different areas of healthcare that are commonly used, and it would help the requirements and analysis process if software engineering principles could adopt these rather than agent abstractions. For example, when 'patient' is so critical a concept to the healthcare domain, modelling them as abstract 'agents' only obfuscates the intention. Similarly, the context in which the clinical researcher operates may be an organisation, but for medics, such organisations are quite distinct from the 'sites', such as clinics or hospitals, from which patients are recruited.

Second, while there are explicit regulations in the domain, there are also many implicit good practices for medicine and healthcare. Capturing these as part of the engineering process is possible but prone to accidental exclusion. It is unclear why these need to be captured every time, and could instead be an embedded part of the methodology. In consequence, in our ongoing work, we are investigating how to address both weaknesses, and provide an AOSE methodology tailored more specifically to e-health.

6 Conclusions

This paper explores the suitability of AOSE techniques for the development of complex systems in the healthcare domain. To investigate this domain, we have designed a real-time system for the identification of eligible patients for clinical trials based on an AOSE methodology. The results obtained show that the use of high level AOSE concepts, such as organisations, roles, norms and contracts, is beneficial to analyse and design health systems. Furthermore, it has been shown that the use of AOSE techniques will produce flexible systems that can deal with the dynamics of the normative and technological environment in the healthcare domain.

Acknowledgment. This work is partially supported by the TIN2009-13839-C03-01, TIN2011-27652-C03-01, CSD2007-00022, COST Action IC0801, FP7-294931 and the FPU grant AP2007-01276

References

1. Argente, E.: GORMAS: Guas para el desarrollo de sistemas multiagente abiertos basados en organizaciones. PhD thesis, Departamento de Sistemas Informaticos y Computacion, Universidad Politecnica de Valencia (2008)
2. Argente, E., Botti, V., Carrascosa, C., Giret, A., Julian, V., Rebollo, M.: An Abstract Architecture for Virtual Organizations: The THOMAS approach. Knowledge and Information Systems, 1–35 (2011)
3. Bajo, J., Fraile, J.A., Pérez-Lancho, B., Corchado, J.M.: The thomas architecture in home care scenarios: A case study. Expert Systems with Applications 37(5), 3986–3999 (2010)
4. Boissier, O., Padget, J., Dignum, V., Lindemann, G., Matson, E., Ossowski, S., Sichman, J.S., Vázquez-Salceda, J. (eds.): ANIREM and OOOP 2005. LNCS (LNAI), vol. 3913. Springer, Heidelberg (2006)
5. Bordini, R.H., Dastani, M., Winikoff, M.: Current issues in multi-agent systems development. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2006. LNCS (LNAI), vol. 4457, pp. 38–61. Springer, Heidelberg (2007)
6. Carabelea, C., Boissier, O.: Coordinating agents in organizations using social commitments. Electronic Notes in Theoretical Computer Science 150(3), 73–91 (2006)
7. Lopes Cardoso, H., Oliveira, E.: A contract model for electronic institutions. In: Sichman, J.S., Padget, J., Ossowski, S., Noriega, P. (eds.) COIN 2007. LNCS (LNAI), vol. 4870, pp. 27–40. Springer, Heidelberg (2008)
8. Chopra, A.K., Dalpiaz, F., Giorgini, P., Mylopoulos, J.: Modeling and reasoning about service-oriented applications via goals and commitments. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 113–128. Springer, Heidelberg (2010)
9. Criado, N., Argente, E., Botti, V.: A normative model for open agent organizations. In: International Conference on Artificial Intelligence, vol. 1, pp. 101–107 (2009)
10. DeLoach, S.A.: Omacs a framework for adaptive, complex systems. In: Handbook of Research on Multi-AGent Systems: Semantics and Dynamics of Organizational Models, pp. 76–104. IGI Global (2009)
11. DeLoach, S.A.: Developing a multiagent conference management system using the O-MaSE process framework. In: Luck, M., Padgham, L. (eds.) AOSE 2007. LNCS, vol. 4951, pp. 168–181. Springer, Heidelberg (2008)
12. DeLoach, S.A., Padgham, L., Perini, A., Susi, A., Thangarajah, J.: Using three aose toolkits to develop a sample design. International Journal Agent-Oriented Software Engineering 3, 416–476 (2009)
13. Dignum, F., Dignum, V., Padget, J., Vázquez-Salceda, J.: Organizing web services to develop dynamic, flexible, distributed systems. In: International Conference on Information Integration and Web-based Applications Services, pp. 225–234 (2009)
14. Dignum, V.: A model for organizational interaction: Based on agents, founded in logic. PhD thesis, Utrecht University (2003)
15. Dignum, V., Meyer, J.-J., Dignum, F., Weigand, H.: Formal specification of interaction in agent societies. In: Hinchey, M.G., Rash, J.L., Truszkowski, W.F., Rouff, C.A., Gordon-Spears, D.F. (eds.) FAABS 2002. LNCS (LNAI), vol. 2699, pp. 37–52. Springer, Heidelberg (2003)
16. Dignum, V., Vázquez-Salceda, J., Dignum, F.: OMNI: Introducing social structure, norms and ontologies into agent organizations. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2004. LNCS (LNAI), vol. 3346, pp. 181–198. Springer, Heidelberg (2005)

17. Dustdar, S., Pichler, R., Savenkov, V., Truong, H.-L.: Quality-aware service-oriented data integration: requirements, state of the art and open challenges. SIGMOD Record Journal 41(1), 11–19 (2012)
18. Esteva, M., Rosell, B., Rodriguez, J.A., Arcos, J.L.: AMELI: An agent-based middleware for electronic institutions. In: International Conference on Autonomous Agents and MultiAgent Systems, pp. 236–243 (2004)
19. Ferber, J., Gutknecht, O., Michel, F.: From Agents to Organizations: An Organizational View of Multi-Agent Systems. In: Giorgini, P., Müller, J.P., Odell, J. (eds.) AOSE 2003. LNCS, vol. 2935, pp. 214–230. Springer, Heidelberg (2004)
20. Ferber, J., Michel, F., Baez, J.: AGRE: Integrating environments with organizations. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2004. LNCS (LNAI), vol. 3374, pp. 48–56. Springer, Heidelberg (2005)
21. Fernández, R.F., Magariño, I.G., Gómez-Sanz, J.J., Pavón, J.: Integration of web services in an agent oriented methodology. Journal International Transactions on Systems Science and Applications 3, 145–161 (2007)
22. Garcia, E., Giret, A., Botti, V.: Software engineering for Service-oriented MAS. In: Klusch, M., Pěchouček, M., Polleres, A. (eds.) CIA 2008. LNCS (LNAI), vol. 5180, pp. 86–100. Springer, Heidelberg (2008)
23. Garcia, E., Giret, A., Botti, V.: Regulated open multi-agent systems based on contracts. In: Information Systems Development, pp. 243–255 (2011)
24. Garcia, E., Giret, A., Botti, V.: Developing Regulated Open Multi-agent Systems. In: International Conference on Agreement Technologies, pp. 12–26 (2012)
25. Gateau, B., Boissier, O., Khadraoui, D., Dubois, E.: Moiseinst: An organizational model for specifying rights and duties of autonomous agents. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2006. LNCS (LNAI), vol. 4389, pp. 41–50. Springer, Heidelberg (2007)
26. Giret, A., Botti, V.: Engineering holonic manufacturing systems. Computers in Industry 60, 428–440 (2009)
27. Gonzalez-Velez, H., Mier, M., Julia-Sape, M., Arvanitis, T., Garcia-Gomez, M.R.J., Lewis, P., Dasmahapatra, S., Dupplaw, D., Peet, A., Arus, C., Celda, B., Huel, S.V., Lluch-Ariet, M.: Healthagents: distributed multi-agent brain tumor diagnosis and prognosis. Applied Intelligence 30 (2009)
28. Gâteau, B., Boissier, O., Khadraoui, D.: Multi-agent-based support for electronic contracting in virtual enterprises. In: IFAC Symposium on Information Control Problems in Manufacturing (INCOM), vol. 150(3), pp. 73–91 (2006)
29. Hermoso, R., Centeno, R., Billhardt, H., Ossowski, S.: Extending virtual organizations to improve trust mechanisms (short paper). In: Proc. 7th INt. Conf. on Autonomous Agents and Multiagent Systems, pp. 1489–1472 (2008)
30. Huhns, M., Singh, M.: Reseach directions for service-oriented multiagent systems. IEEE Internet Computing, Service-Oriented Computing Track 9(1) (2005)
31. Isern, D., Sánchez, D., Moreno, A.: Organizational structures supported by agent-oriented methodologies. Journal of Systems and Software 84(2), 169–184 (2011)
32. Kurtanovic, Z., Schumann, R., Timm, I.J.: Model-driven specification of strategies for negotiating agents. In: Proceedings of the 13th International Workshop on Agent-Oriented Software Engineering (AOSE 2012) held at AAMAS 2012 (2012)
33. Lin, C.-E., Kavi, K.M., Sheldon, F.T., Daley, K.M., Abercrombie, R.K.: A methodology to evaluate agent oriented software engineering techniques. In: Hawaii International Conference on System Sciences, p. 60 (2007)
34. Meneguzzi, F., Modgil, S., Oren, N., Miles, S., Luck, M., Faci, N.: Applying electronic contracting to the aerospace aftercare domain. Engineering Applications of Artificial Intelligence 25(7), 1471–1487 (2012)

35. Mora, D., Taisch, M., Colombo, A.W., Mendes, J.M.: Service-oriented architecture approach for industrial system of systems: State-of-the-art for energy management. In: 2012 10th IEEE International Conference on Industrial Informatics (INDIN), pp. 1246–1251 (2012)
36. Oren, N., Panagiotidi, S., Vázquez-Salceda, J., Modgil, S., Luck, M., Miles, S.: Towards a formalisation of electronic contracting environments. In: Hübner, J.F., Matson, E., Boissier, O., Dignum, V. (eds.) COIN@AAMAS 2008. LNCS, vol. 5428, pp. 156–171. Springer, Heidelberg (2009)
37. Paranjape, R., Sadanand, A.: Multi-Agent Systems for Healthcare Simulation and Modeling: Applications for System Improvement. Information Science Reference - Imprint of: IGI Publishing (2009)
38. Pavon, J., Gomez-Sanz, J., Fuentes, R.: The ingenias methodology and tools. In: Agent-Oriented Methodologies, ch. IX, pp. 236–276. Henderson-Sellers (2005)
39. Rammal, A., Trouilhet, S., Singer, N., Pecatte, J.-M.: An adaptive system for home monitoring using a multiagent classification of patterns. In: International Conference on Business Process Management, pp. 3:1–3:8 (2008)
40. Singh, M.P.: Commitments in multiagent systems: Some history, some confusions, some controversies, some prospects. In: The Goals of Cognition. Essays in Honor of Cristiano Castelfranchi, pp. 1–29. College Publications (2011)
41. Taweel, A., Delaney, B., Speedie, S.: Towards achieving semantic interoperability in ehealth services. In: Watfa, M. (ed.) E-Healthcare Systems and Wireless Communications: Current and Future Challenges, pp. 388–401. IGI (2012)
42. Taweel, A., Speedie, S., Tyson, G., Tawil, A.R., Peterson, K., Delaney, B.: Service and model-driven dynamic integration of health data. In: International Workshop on Managing Interoperability and Complexity in Health Systems, pp. 11–17. ACM (2011)
43. Telang, P.R., Singh, M.P.: Enhancing Tropos with Commitments. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications. LNCS, vol. 5600, pp. 417–435. Springer, Heidelberg (2009)
44. Telang, P.R., Singh, M.P.: Comma: A commitment-based business modeling methodology and its empirical evaluation. In: International Conference on Autonomous Agents and MultiAgent Systems, pp. 1073–1080. IFAAMAS (2012)
45. Trencansky, I., Cervenka, R.: Agent modelling language (AML): A comprehensive approach to modelling mas. Informatica 29(4), 391–400 (2005)
46. Tyson, G., Taweel, A., Miles, S., Luck, M., Van Staa, T., Delaney, B.: An agent-based approach to real-time patient identification for clinical trials. In: Kostkova, P., Szomszor, M., Fowler, D. (eds.) eHealth 2011. LNICST, vol. 91, pp. 138–145. Springer, Heidelberg (2012)
47. Tyson, G., Taweel, A., Zschaler, S., Van Staa, T., Delaney, B.: A model-driven approach to interoperability and integration in systems of systems. In: Modelling Foundations and Applications Workshop (2011)
48. Vecht, B., Dignum, F., Meyer, J.-J., Dignum, M.: Handbook of research on multi-agent systems: Semantics and dynamics of organizational models. In: Autonomous Agents Adopting Organizational Rules, pp. 314–333. IGI Global (2009)
49. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: The gaia methodology. ACM Transactions on Software Engineering Methodology 12, 317–370 (2003)
50. Zhang, X., Xu, H., Shrestha, B.: Building a health care multi-agent simulation system with role-based modeling. In: MAS for Health Care Simulation and Modeling: Applications for System Improvement, ch. VI. IGI Global (2009)

How to Extract Fragments from Agent Oriented Design Processes

Valeria Seidita[1,2], Massimo Cossentino[2], and Antonio Chella[1]

[1] Dipartimento di Ingegneria Chimica, Gestionale, Informatica, Meccanica, University of Palermo, Italy
{valeria.seidita,antonio.chella}@unipa.it
[2] Istituto di Reti e Calcolo ad Alte Prestazioni, Consiglio Nazionale delle Ricerche, Palermo, Italy
cossentino@pa.icar.cnr.it

Abstract. Using Method Engineering for creating agent oriented design processes is a challenging task because of the lack of a fragment repository defined and filled starting from a shared and unique definition of fragment. The creation of a repository implies the fragmentation of existing agent design processes. In this paper we propose a set of guidelines for extracting fragments from agent design processes. The work is based on a precise definition of fragment and it aims to establish a method for fragmenting processes and obtaining homogeneous fragments regardless of how the starting design processes are defined and described.

1 Introduction

When method designer is about to create a new agent oriented design process using a (Situational) Method Engineering [5] compliant approach she needs a repository of fragments; she needs to inspect and query the repository in order to quickly and easily select the fragments that best fit her needs and that she may assemble in order to compose/create the new agent design process. If the fragments are correctly identified the result of the assembly phase would be the design process that best allows to develop the multi-agent system solving a specific class of problems.

Currently there are several very good approaches to Situational Method Engineering (SME) [20][7][21][15], all of them primarily aim to re-use components that come from other existing design processes, hence they provide solutions already used and tested. Nevertheless each approach follows its own logic, based on the specific used definition of fragment, and as a consequence the related repository has very specific features that make it little reusable. A repository is created starting from the fragmentation of existing design processes and currently all the existing ones are created in a naive fashion, often based on the skills and experience of the method designer. There is not a well-specified technique to split existing agent design processes and to extract fragments from them.

The fragmentation process basically prescribes to identify some points in which "to cut" the design process. The problems we are facing are: how to find these points? How to understand and verify that the identified pieces have a meaning, are consistent, from a methodological point of view? It is to be remembered in fact that the fragment of a process is a process itself, therefore it should be a component with a meaning in the process perspective; in our case in an agent oriented process perspective. All this has not found adequate answers so far because if the method engineer wants to extract fragments she must have a clear idea of what is a process fragment for an agent oriented process and she has to highlight the main elements of the fragment that must be sought in the design process. We can see the fragment as a piece of a puzzle, if we do not know how the piece of a puzzle is done we cannot find it in the puzzle, we would always see the puzzle as a whole.

A first and fairly established definition of fragment already exist [9]; on the basis of this definition several research groups broke down their own agent design process into pieces and extracted several fragments that today form a first prototype of agent fragment repository[1]. Despite that, each fragmentation process is not replicable and repeatable on another agent design process and above all it can be done only by (or with the tight assistance of) the design process experts. Moreover, the fragmentation obtained without the aid of specific guidelines has led to fragments that are different in granularity and in the representation. They are not homogeneous even if they are based on the same definition; this fact makes their assembly very difficult.

What we propose in this work is a mean for answering to the previous reported questions. The first question is met by considering the improved definition of fragment we introduce in this paper; above all the granularity of the fragment and the work product lead to the identification of what we call the "*cutting points*". The granularity and the work product are strictly related to our definition of fragment. The answer to the second question is guaranteed by the use of the multi-agent system (MAS) metamodel during the extraction process. The multi-agent system metamodel is the core of the fragment definition and it is one of the most important concepts in agent oriented software engineering. Moreover, MAS metamodel strongly distinguishes our approach from the others by providing the key point for applying our SME approach.

In the remainder of this paper, we first illustrate the background on (situational) method engineering and the motivations of our work (section 2). In section 3 we define the concept of process fragment that is the basis of our approach on SME and in section 4 we introduce the guidelines for fragmentation also reporting examples of application to an agent design process and a classical design process. Finally in sections 5 and 6 some discussions and conclusions are drawn.

[1] In `http://www.pa.icar.cnr.it/passi/` and in `http://www.pa.icar.cnr.it/cossentino/fragrep/` fragments from PASSI [8], Tropos [4] and Adelfe [3] can be found. Fragments from INGENIAS [22] can be found in `http://grasia.fdi.ucm.es/main/fr/node/241`.

2 Background and Motivation

Method Engineering is the "engineering discipline to design, construct and adapt methods, techniques and tools for the development of systems", this is the definition generally accepted since [5]. Method Engineering (ME) and Situational Method Engineering (SME), which is the part of ME dealing with the creation of method(ologies) for specific situations, is the answer to the historical problem of the lack of a one-size-fits-all methodology [20][28][27].

Several approaches to situational method engineering are present in literature, they all descend from the assumptions made by Brinkkemper in [5] and then by Gupta and Prakash in [14] that a method(ology) engineering process is composed of three main phases: method requirements engineering, method design and method construction. More in details Brinkkemper highlights the following steps: characterization of the project, selection of the method fragments, assembly of the method fragments. The method fragment is a piece of existing method(ology), an optimized method(ology) for specific situation may be constructed by reusing relevant method fragments.

Literature emphasized two different modus operandi for applying situational method engineering; one is called assembly based method engineering, the other may be called method engineering by configuration. The assembly based method engineering mainly focuses on assembling method fragments. Recently Ralyté et al. [23] developed a generic process model for situational method engineering. This is an assembly based process model implying the specification of method requirements, the selection of method fragments and finally the assembly of method fragments. Method engineering by configuration prescribes to adapt one particular method to different situations [19].

In any of the previous cases situational method engineering does not principally focus on obtaining method fragments but all the approaches assume to have an already filled repository from which to select method fragments. Experiments of fragmentation and of repository creation have been made in (see [11][12]) but none of them provide guidelines for fragmenting, infact these repositories are only based on the experiences made that in most cases are not repetable.

We think that we cannot be exempt from considering the criticality of this part of SME. How might we create a repository of fragments to be used for creating specific agent oriented design processes? Therefore the lack of an agent oriented fragment repository, structured in a way that makes easy reusing fragments and fosters the interoperability of the existing approaches, severely limits and affects the use of the SME. It is in fact still considered too difficult to apply.

Having guidelines for fragmentation is very important because one way to break up can lead to homogeneous fragments in which the interfaces are easily identified; this can make the selection of the fragments and their assembly faster and easier. This problem is difficult to tackle because there are many agent oriented design processes from which we can extract fragments but each of them presents features strongly tied to the type of multi-agent system they are devoted to develop. Often there is no proper or standard documentation, sometimes the elements that are highlighted in one design process are not dealt with in another one or have different definitions. The substantial difference between design processes, which reflects the intrinsic difference of the different types of agents treated, makes fragmentation very difficult; especially if this process needs to be done by a person who is not fully aware of every detail of the design process.

The fragmentation method we propose overcomes this limitation because of the use we make of an important element of agent design processes, the multi agent system metamodel. In our approach the system metamodel is a fundamental part of the fragment definition and it is also the core for fragments selection, for assembling them [26] and, as it will be illustrated later, for extracting fragments from existing design processes.

3 The Adopted Fragment Definition

The most important concept in the Situational Method Engineering approach is the *method fragment*, thus it is mandatory to have a repository of fragments and techniques for selecting and assembling them. Moreover a relevant part of the method designers' work concerning the population of fragments repository notably influences the whole SME process. From the experiences made in constructing agent design processes we learnt that the techniques for selecting and assembling fragments, for extracting them from existing design processes and then for storing in the repository strongly depend on the definition of fragment the method designer uses during his work.

Although the focus of this paper is on the fragmentation guidelines, it is important to introduce the definition of fragment that forms the basis of our approach to SME. We already defined the concept of fragment [9] (from now on it will be referred to as *process fragment* for reasons that will be clear later in the section), and now we give an overview on the process fragment definition and the improvement introduced to the first version. The notions here introduced are necessary for understanding the guidelines, how they were created and why some steps are present.

We agree on the general statement that the process fragment is a reusable portion of a design process. Let us dwell on the terms "reusable" and "design process": reusable, we want to create a mean for the method engineer to quickly and easily select and then reuse process fragments for creating new agent design processes. Design process, Fuggetta in [13] defined the software development design process as *"the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy and maintain (evolve) a software product"*.

We adopt this definition of design process. In plain terms, we could say that design process includes components devoted to make explicit what is the work to be done for pursuing an objective, hence delivering some kind of artefact, who has to do this work and how. The process fragment may be seen as a complete portion of a design process constructed in such a way it can be profitably (and rather easily) reused in a new design process creation. The name we chose to adopt (process fragment) is a direct consequence of that. Most commonly in literature the reader can find names like method fragment or chunk. As it will be evident later on, the proposed definition of process fragment is near to the chunk concept [2][21], while the method fragment definition [20][6][17] addresses a smaller item, pragmatically speaking a process fragment can be built by assembling several method fragments. Up to now, it can be said that design process, and process fragment, ground on three principal components: activities, artefacts or work products and stakeholders or roles.

The principal difference of our approach against the others is that we add to this triad another very important concept that is crucial in the agent oriented software engineering, the *multi-agent system metamodel*. The system metamodel is the representation of constructs needed for creating system models.

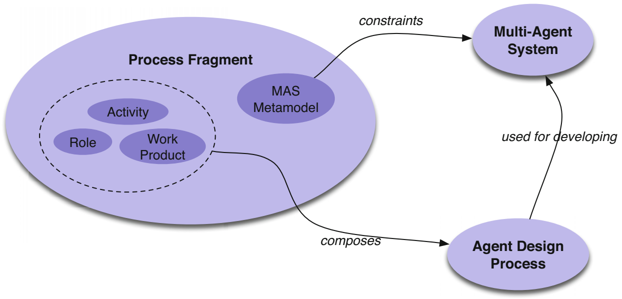


Fig. 1. An Overview on Process Fragment

Fig. 1 shows an overview of the process fragment concept and its use in our approach. The process fragments compose an agent oriented design process and are mainly composed of activities, work products, roles, and multi-agent system (MAS) metamodel. The MAS metamodel contains constructs to be instantiated during the design activities and constraints the multi-agent system. In the following a detailed list of all the elements of our process fragment definition is reported:

– specification of the workflow, the work to be done (activities, tasks or steps[2]), the roles performing it and the work products produced;
– a list of constructs of the multi-agent system metamodel to the defined (or refined) through the fragment workflow;
– a description of the fragment goal, for providing the reader with a quick understanding of the design objective pursued by the process fragment workflow; for instance "the aim of this fragment is to identify the agents involved in the system";

[2] See SPEM 2.0 specification [1] for a definition of these items.

- a description of the fragment origin and its granularity. This part lets the method designer have a quick idea on the focus and the domain in which the fragment might work, this can also allow a sort of automatic or semiautomatic selection of the fragments;
- a set of guidelines that are divided in enactment guidelines and reuse guidelines, the first helps the designer while she is using the fragment for producing a portion of the multi-agent system, the second is used by the method designer while creating a new design process for having means for carrying on the selection and the assembly phases;
- a glossary of terms used in the fragment; this prevents misunderstandings if the fragment is reused in a context that is different from the original one.

As regard the granularity, our SME approach does not require to establish a *length* for the portion of process in the process fragment; it is only needed that it can manage some specific elements. Therefore we can have process fragments at three different levels of granularity: phase fragment, composed fragment and atomic fragment as defined below.

Phase Process Fragment. A phase process fragment delivers a set of work products belonging to the same design abstraction level of the design flow. An examples of phase-level work product may be a system analysis document; it is composed of several work products (diagrams, text documents, ...) all belonging to the same design abstraction level (system analysis).

Composed Process Fragment. A composed (process) fragment delivers a work product (or a set of instances of the same work product). Composed process fragments may be nested. This is an obvious choice to allow the maximum flexibility for representing whatever size of fragment. For instance a composed fragment delivering a composite work product may be composed by composed fragments delivering non-composite work products (free text, structured text, diagram,...) or even atomic fragments (see below). An example of composed fragment may consists in a portion of a process where the designer models use cases. This fragment delivers a work product (use case diagrams and a description text document) that is part of the System Analysis document produced by the System Analysis phase fragment.

Atomic Fragment. An atomic (process) fragment delivers a portion of a work product and/or a set of system model constructs (in terms of their instantiation or refinement). A portion of a work product is here intended never to be a whole work product; in other words, atomic fragments never deliver entire work products. An atomic fragment may also not deliver a portion of work product but rather it may deliver a portion of the system model. An example of atomic fragment may be the identification of actors to be used for modeling use cases by starting from the analysis of some text describing system behavior.

Finally, as regard the MAS metamodel we performed an extended experimentation with some existing agent design processes. We realized that different types of metamodel may also be considered from the point of view of the set of constructs included in them:

Complete System Metamodel. It includes all the system metamodel constructs (elements and relationships) that are managed by the designers using a specific design process. This also includes all the constructs that are accepted as external inputs in the process.

Definable System Metamodel. It includes all the system metamodel constructs that are instantiated during the design process enactment. This is a subset of the complete system metamodel.

Workproduct System Metamodel. It only includes all the complete system metamodel constructs that are reported and drawn in the design process work products.

4 The Proposed Fragmentation Approach

Basing on the previous definition of process fragment and on the assumption that the MAS metamodel has a central role in designing multi-agent systems, we established guidelines for extracting process fragments from existing design processes. Let us suppose to have an agent design process anyway described and documented; it is not important if we have text documentation, produced in a formal way, or an oral representation of the design process. What is important is to have a complete description of the process in form of activities to be performed, work products to be delivered, stakeholders to be involved, the system metamodel to be instantiated and enactment guidelines (illustrating how to perform activities and to produce work products). These latter are often referred to as techniques.

The key idea is to consider the design process as a workflow of activities which bring to some kind of results; this can be done both with a mere sequential design process and an iterative and incremental one. In both the cases we can see the whole process as a temporal line where at each time t_i a portion of work (a set of activities) produces an outcome, a work product, a diagram or an entire model (see Fig. 2).

The first step in the fragmentation process consists in analyzing the whole design process. This activity let us identify a set of portions of the process. Each portion of the process is composed of activities that begin at the time t_{n-1} and end at the time t_n, when it can be thought that a specific identifiable portion of the system has been designed; the time t_n is one of the cutting points we may identify.

This is the general rule for recognized cutting points; going into more details, the cutting points depend on the granularity of process fragment we want to extract, hence three different case may be pointed out:

– while extracting phase process fragment - the time t_i occurs when we can identify a set of work products representing a complete model of the system;
– while extracting composed process fragment - the time t_i occurs when we can identify a complete work product modeling a portion of the system;

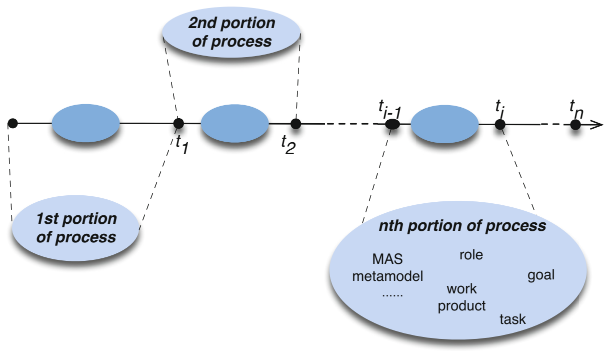


Fig. 2. The Design Process seen as a Temporal Line

- while extracting atomic process fragment - the time t_i occurs when during the hypothetical enactment process we complete the definition or the refinement of one element of the system model or we produce a consistent portion of a work product.

Once the cutting point has been identified the portion of process it delimits has to be investigated in order to identify the main elements that constitutes process fragment. The steps we ought to follow include:

1. identifying the main outcome, likely a work product or a portion of it or the definition of a MAS metamodel construct for atomic fragments.;
2. identifying the portion of the multi-agent system metamodel which elements are instantiated in the portion of work;
3. identifying all the elements (work product and/or MAS metamodel constructs) needed as input for carrying on the selected portion of work;
4. identifying for this portion of work all the elements that will form the process fragment: roles, tasks, goals of the fragment, etc.

4.1 The Details about the Guidelines

Fig. 3 sketches the work the method designer does when she uses our guidelines. In order to better illustrate the guidelines let us follow an example for their application. Suppose the method designer wants to extract process fragments from the PASSI design process [8] and that she chooses the composed level of granularity.

PASSI (Process for Agent Societies Specification and Implementation) [8] is an agent oriented design process for designing peer-agents and covers all the phases from the requirements analysis to the code of the multi-agent system. PASSI includes five phases arranged in an iterative/incremental process mode. Each phase produces a document that is usually composed aggregating UML

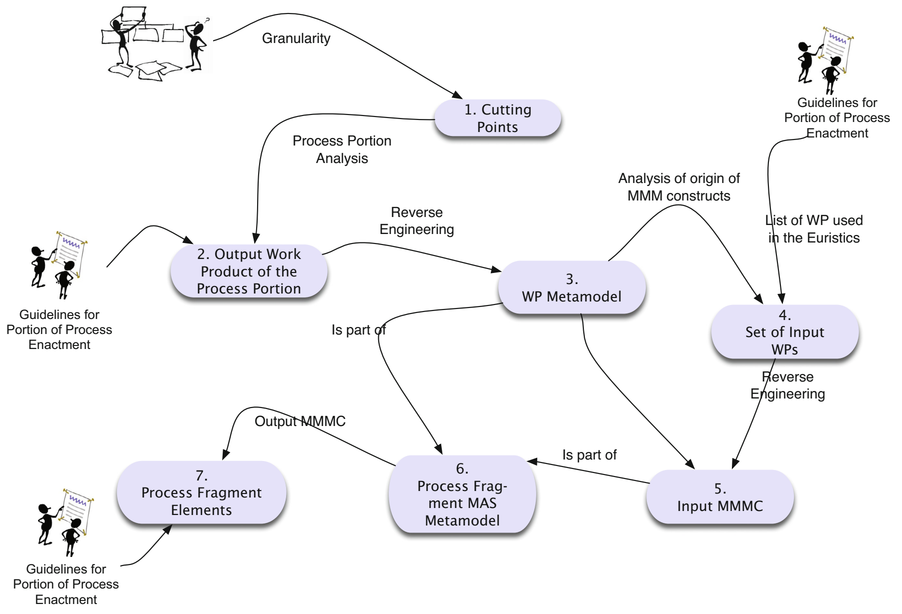

Granularity
1. Cutting Points
Process Portion Analysis
Guidelines for Portion of Process Enactment
2. Output Work Product of the Process Portion
Reverse Engineering
Analysis of origin of MMM constructs
List of WP used in the Euristics
3. WP Metamodel
4. Set of Input WPs
Is part of
Reverse Engineering
Output MMMC
Is part of
7. Process Fragment Elements
6. Process Fragment MAS Metamodel
5. Input MMMC
Guidelines for Portion of Process Enactment
Guidelines for Portion of Process Enactment

Fig. 3. The Guidelines for the Fragmentation Process

models, text descriptions, tables, code, and so on produced during the related activities. Each phase is composed of one or more activities each one responsible for designing or refining one or more artefacts that are part of the corresponding model. For instance, the System Requirements model includes an agent identification diagram that is a kind of UML use case diagrams and also some text documents like a glossary and the system user scenarios.

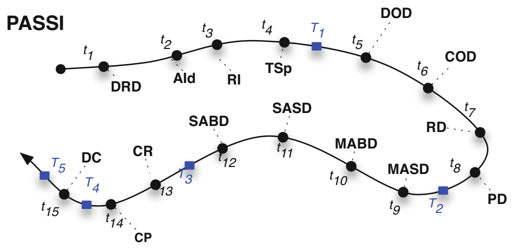
After a first analysis the method designer may identify fifteen cutting points (remember that she is looking for composed process fragments), each of them corresponds to the time t_i when a work product is delivered (see Fig. 4).

Starting from the beginning, consider the portion of process before the time t_1, the work product of this first portion is the *Domain Requirements Description (DRD) diagram*, a functional description of the multi-agent system using UML use case diagrams with their textual description.

The method engineer analyzes this work product (STEP 2. OUTPUT WORK PRODUCT OF THE PROCESS PORTION in Fig. 3) and through a reverse engineering process identifies the MAS metamodel construct (MMMC) here reported.

This activity is very simple when the work product is only in form of diagrams, in fact each graphical element of the diagram corresponds to one construct of the metamodel, the only thing to do is finding the mutual relationships among constructs. This descends from the assumption (section 3) we made about metamodeling: every element in the whole model of the system, in a work product or in a document specifying the system, is an instance of one construct of the MAS metamodel.

In the case of the DRD, when we draw the use case we instantiate a functional requirement, when we draw the “actor" we instantiate an actor and the associations are instance of generalize, include, extend and association relations.



DRD: Domain Requirements Descr.
AId: Agents Identification
RId: Roles Identification
TSp: Task Specification
DOD: Domain Ontology Descr.
COD: Communication Ontological Descr.
RD: Roles Description
PD: Protocol Description
MASD: Multi Agent Structure Descr.
MABD: Multi Agent Behaviour Descr.
SASD: Single Agent Structure Descr.
SABD: Single Agent Behaviour Descr.
CR: Code Reuse
CP: Code Production
DC: Deployment Configuration

Fig. 4. The Cutting Points for PASSI

Figures 5.a) and 5.b) show an excerpt of the DRD diagram and the related work product MAS metamodel (Step 3. WP Metamodel).

The next step is to identify all the WPs that serve as inputs for carrying on the work in this first portion of process. In performing this activity the method engineer needs to consult the design process enactment guidelines. Roughly speaking, the work done during a process activity aims to define elements that the designer reports in the work product, for doing that she makes some kind of reasoning, above all on other elements of the system domain. For instance, in order to elicit requirements of the system, the analyst often uses textual scenarios for gaining the interaction between the system and the users and the related functional requirements. This is the case of the DRD, by looking at the enactment guidelines the method engineer may deduce that the elements in the DRD work product are designed by managing the concept of *Scenario*.

Hence the set of input WPs (Step 4.Set of Input WPs) includes two text documents, the *Problem Statement* and *Scenarios*; the input MAS metamodel is only composed of the construct *User Scenario* (Step 5. Input MMMC). The constructs of the input MAS metamodel are identified in the same way of the step 3, through a reverse engineering process. Often, part of the constructs of the input MAS metamodel are reported, hence quoted, in the work product of the portion of work.

At this point the method designer has all the elements useful for establishing the complete MAS metamodel (Step 6. Process Fragment MAS Metamodel), indeed it is the sum of the previous two ones. In the case of our example the complete MAS metamodel is shown in Fig. 6.

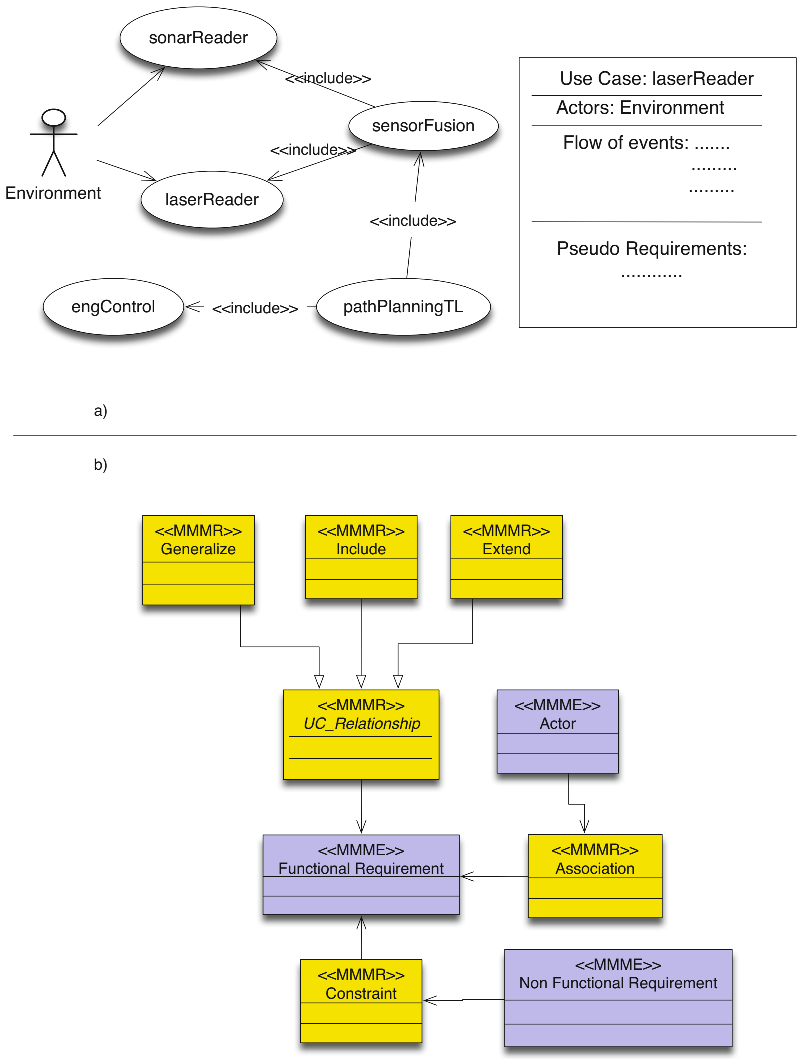


Fig. 5. a) An Excerpt of the DRD Work Product; b) The Work Product MAS Metamodel

After the sixth step, the extraction of the first composed process fragment is complete, the method designer must now (Step 7. Process Fragment Elements) identify the other elements in the fragment; first of all activities, roles and work products and then all the others such as goals, reuse guidelines, and so on. Now that the first fragment has been identified and extracted the method designer may consider the rest of design process and she may extract all the other composed fragments.

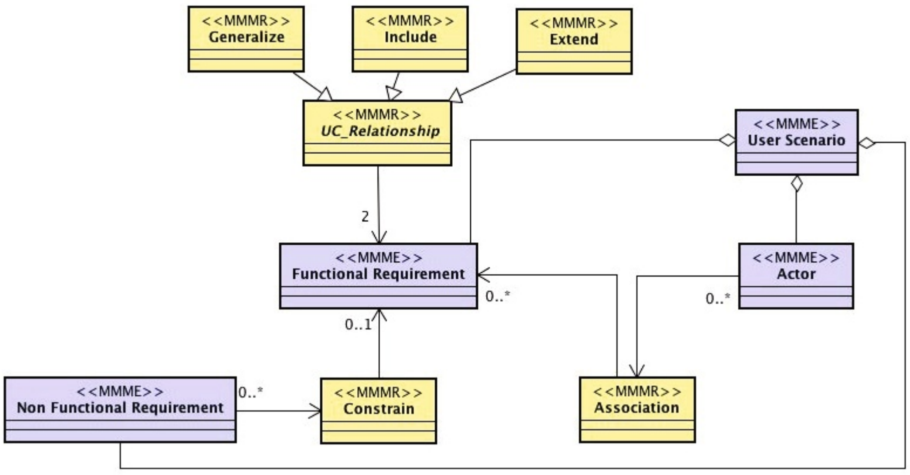


Fig. 6. The Complete MAS Metamodel for the DRD Process Portion

The same guidelines may be applied for extracting phase or atomic fragments; in the case of a phase fragment for PASSI, the time T_1 (see Fig. 4) is when the set of work products dealing with the problem domain description has been delivered; from the definition a phase fragment is the one delivering a set of work products that belong to the same design abstraction level.

In the PASSI example the method designer can find five different cutting points, they are illustrated in capital letter in the Fig. 4 and correspond to the five main phases PASSI is composed of.

Once the method designer has found the cutting points, she can perform the same activities done for obtaining the composed process fragments. The number of work products to analyze is greater than one in fact one phase fragment can be considered a composition of composed process fragments.

Some differences are present when the method designer wants to extract an atomic fragment. An atomic fragment is different from the other two in the outcome, it has been conceived for representing the smaller piece of work the designer may perform. For instance all those situations in which the designer makes some kind of reasoning, sometimes without producing concrete outcome and following for instance some heuristics, for designing a specific MAS metamodel construct. Other times atomic fragments deliver portions of work products.

The cutting points can be identified following the process workflow and stopping it when a portion of work product is completed, in the sense that one element of the system model is completely defined. For instance, if a work product aims at defining the agents involved in the system and their goals, a part of the work product may be the one listing or representing all the agents, supposing that the flow of work for producing that work product implies firstly to find all the involved agents and then to identify their goals.

In the PASSI example, the first part of the design process prescribes that the system analyst and the domain expert collaborates in identifying use cases and then they describe them in order to produce the previous said DRD diagram. The first part of the work does not imperatively result in a work product, the analyst and the designer might list the use cases in a document or they might keep them in their mind and then represent them in the diagram (in a following portion of work that will be part of another atomic fragment).

Whatever is the result, the work done aims to instantiate several times the functional requirement construct in order to obtain use cases; this portion of work can represent an atomic fragment and has to be described and represented as the others two.

We claim that the proposed guidelines can be applied for extracting process fragments from agent design processes also described in an informal or not well structured way; the work to be done in this cases could be a little more demanding but it leads anyway to good results. The best result is obtained if the agent design processes were documented using the FIPA IEEE standard template [18].

This specification requires to emphasize the main elements of design process we shown in Fig. 1 by means of SPEM [1] activity and class diagrams thus providing dynamic and structural views of the design process parts, metamodels, tables for representing input/output workproduct, input/output MAS metamodel constructs and other elements that allow a quick and a complete view on the overall design process. Using this kind of representation leads to a very quick and easy application of the guidelines since the cutting points can be rapidly identified and then all the information is visible and identifiable[3].

It is worth to note that basing on our experience the best way for extracting fragments from a design process is using a top-down approach, hence we suggest starting with the extraction of all the phase process fragments and then the composed process fragments.

4.2 Applying the Guidelines to OpenUP

The proposed guidelines was born in the context of agent oriented design processes but the system metamodel concepts we use led us to consider the possibility of applying our approach to all kinds of design process. In the last months, in collaboration with a great part of the agent research community, we worked in representing some agent design processes using the IEEE-FIPA standard documentation template [18]. During this experience we also documented the OpenUp (*http://epf.eclipse.org/wikis/openup/*) design process and we obtained good results in terms of the applicability of the standard.

OpenUP is an iterative process composed of four main phases: Inception, Elaboration, Construction, and Transition, through which each iteration is performed. Depending on a series of factors like, for instance, the technology used, the architectural complexity and the project size, each phase may present many

[3] The specification can be found in
`http://fipa.org/specs/fipa00097/index.html`

iterations of a variable length. OpenUP also provides work breakdown structure (WBS) templates for each iteration, and a WBS template for an end-to-end process. Applying the standard allowed us to identify, for each phase, all the activities, the roles, the work products and the complete system metamodel of the overall process.

Afterwards, we tried to apply the guidelines for fragmentation to OpenUp. We firstly identified the phase process fragments and then all the composed process fragment; for each of them the applicability of steps 3, 4 and 5 of the guidelines were guaranteed by the complete description of the activity work flow provided by the standard documentation. OpenUp has a peculiarity, it is an iterative and incremental design process; during each phase the same activities are performed each time with a different emphasis over the course of the project. Given this fact, we found in a particular situation, we extracted the first phase process fragment, the *Inception Fragment*, and all its inner composed process fragments (see http://www.pa.icar.cnr.it/cossentino/fragrep/ for a tentative list of OpenUp process fragments), then we identified the second phase fragment, *Elaboration Fragment*, where the most part of the workflow perfectly follow the first phase process fragment workflow, only a little part is refined. Going on to the composed process fragments we found a lot of fragments exactly alike to the composed fragments of the first phase and only some new more.

The same happened for the other phases, until the end of the whole OpenUP; this is due to the fact that OpenUp is iterative and incremental. Therefore, two considerations, in this case the extraction of phase fragments goes out the window and it makes sense to extract all the composed process fragments in their most complete form, as we had extracted at their latest iteration. It is up to the method designer, during the process composition phase, to cut and adapt them to his needs and to the features of the process she is creating. What is important is that we could populate our repository with fragments coming from OpenUp by using our guidelines that revealed to be usefully applicable also to a not agent, iterative and incremental design processes.

5 Discussions

The guidelines for extracting process fragments we illustrated in the previous section are only a part of the whole approach to Situational Method Engineering we developed; it includes the creation of the repository [25], the formalization of a SME process phases (selection, retrieval and assembly of fragments) [26] and, the most important, the definition of the process fragment, what it is composed of and the best way to document it [9][24]. These guidelines was created in a way that lets the method engineer have all the elements for documenting the process fragments and respect the improved definition we gave in this paper and the template we proposed in [24]. Documenting the process fragment using this template aims to mainly put to evidence the process and the product part of the fragment.

One of the major advantages introduced by the proposed method for fragmenting agent design processes, and then extracting process fragments from them, is simplifying the part of situational method engineering devoted to the creation of fragment repositories. Until now, this topic has not been widely considered, indeed commonly the work of the method designer is thought to start from an already filled repository. It is worth to note that the proposed method fragment definition gives the possibility of creating an agent fragments repository where all the fragments are homogeneous because they share the same definition and above all they are structured basing on a core concept, the multi-agent system metamodel for which in [10] we illustrated how it is constructed and all the rules for extracting knowledge on the process.

Moreover, we decided to apply the fragmentation guidelines to OpenUp in order to test our approach upon a "not-agent" oriented design process and above all upon a design process we are not skilled to; we well know PASSI, we used it in several projects and our level of knowledge could have twisted the results on the applicability of the guidelines. For sure, having OpenUp described in the standard way IEEE-FIPA prescribes was of great help but we can say that the applicability of our guidelines is guaranteed by the particular process fragment definition we use. Without the standard description we would have spent more time in identifying the main elements fragment is composed of but the result would be the same. The fragment definition underlying the guidelines together with looking at the design process workflow let us easily retrieve all the fragment elements.

With this work we, now, have all the ingredients for increasing the fragment reusability, hence selection and then assembly and for establishing the basis of a formal approach to SME for (agent) design process creation. Besides having a well specified set of data regarding the process fragment (the metamodel, the activities, the role and the work products, all represented in a well defined fashion) gives the possibility of automatically supporting the overall SME approach and making some kind of reasoning that might lead, for instance, to automatic selection or assembly of fragments.

6 Conclusions and Future Works

In this paper we propose guidelines for fragmenting and then extracting fragments from agent oriented design processes. The guidelines are based on a definition of process fragment that has been improved from the previous one [9].

Our aim is to handle a part of the Situational Method Engineering that, in all the existing approaches, has not been treated with the right details. Indeed, in most cases, applying a SME approach means to start from an already filled repository; how to construct the repository has not been sufficiently investigated yet. Every existing SME approach implies a selection phase from a repository, we claim that if this one is not realized in such a way that the fragments were homogeneous and with easily identifiable interfaces, it would be very difficult to select and to assemble the right fragments meeting the method engineer needs.

It is anyway difficult to extract process fragments from existing agent oriented design processes because of the lack of a proper documentation highlighting the elements characterizing the process fragment, they are: activities, roles and work product, hence the work to be done for obtaining a design result and the stakeholder performing the work. A fourth element, very important in our fragment definition, is the multi-agent system metamodel; by using it we are able to overcome all the limitations said before and to provide guidelines for extracting fragments thus obtaining process fragments of the same granularity.

In the future we plan to develop a tool for aiding the method designer in automatically (or semi-automatically) selecting and assembling process fragments; this will be done exploiting the structure of the fragments and the fact that a lot of important information inside the process can be inferred by means of the system metamodel.

References

1. Software Process Engineering Metamodel. Version 2.0. Final Adopted Specification ptc/07-03-03
2. Backlund, P., Ralyté, J., Jeusfeld, M., Kühn, H., Arni-Bloch, N., Goossenaerts, J., Lillehagen, F.: An interoperability classification framework for method chunk repositories. In: Advances in Information Systems Development, pp. 153–166 (2007)
3. Bernon, C., Camps, V., Gleizes, M.-P., Picard, G.: Engineering adaptive multi-agent systems: the adelfe methodology. In: Agent Oriented Methodologies, ch. VII, pp. 172–202. Idea Group Publishing (2005)
4. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: Tropos: An agent-oriented software development methodology. Autonomous Agent and Multi-Agent Systems 3(8), 203–236 (2004)
5. Brinkkemper, S.: Method engineering: engineering the information systems development methods and tools. Information and Software Technology 37(11) (1996)
6. Brinkkemper, S., Lyytinen, K., Welke, R.: Method engineering: Principles of method construction and tool support. International Federational for Information Processing 65, 336 (1996)
7. Brinkkemper, S., Welke, R., Lyytinen, K.: Method Engineering: Principles of Method Construction and Tool Support. Springer (1996)
8. M. Cossentino. From requirements to code with the PASSI methodology. In: Agent Oriented Methodologies [16], ch. IV, pp. 79–106
9. Cossentino, M., Gaglio, S., Garro, A., Seidita, V.: Method fragments for agent design methodologies: from standardisation to research. International Journal of Agent-Oriented Software Engineering (IJAOSE) 1(1), 91–121 (2007)
10. Cossentino, M., Seidita, V.: Metamodeling: Representing and modeling system knowledge in design processes. In: Proceedings of the 10th European Workshop on Multi-Agent Systems, EUMAS 2012, pp. 103–117 (2012)
11. Esfahani, H.C., Yu, E.: A repository of agile method fragments. In: Münch, J., Yang, Y., Schäfer, W. (eds.) ICSP 2010. LNCS, vol. 6195, pp. 163–174. Springer, Heidelberg (2010)
12. Firesmith, D., Henderson-Sellers, B.: The OPEN Process Framework: An Introduction. Addison-Wesley (2002)
13. Fuggetta, A.: Softaware process: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering, Limerick (Ireland), June 4-11, pp. 25–34. ACM Press, New York (2000)

14. Gupta, D., Prakash, N.: Engineering Methods from Method Requirements Specifications. Requirements Engineering 6(3), 135–160 (2001)
15. Henderson-Sellers, B.: Method engineering: Theory and practice. In: Karagiannis, D., Mayr, H.C. (eds.) Information Systems Technology and its Applications, pp. 13–23 (2006)
16. Henderson-Sellers, B., Giorgini, P.: Agent Oriented Methodologies. Idea Group Publishing, Hershey (2005)
17. Henderson-Sellers, B., Ralyté, J.: Situational method engineering: State-of-the-art review. J. UCS 16(3), 424–478 (2010)
18. IEEE Foundation for Intelligent Physical Agents. Design Process Documentation Template, Document number XC00097A-Experimental (2011)
19. Karlsson, F., Agerfalk, P.: Method configuration: adapting to situational characteristics while creating reusable assets. Information and Software Technology 46(9), 619–633 (2004)
20. Kumar, K., Welke, R.: Methodology engineering: a proposal for situation-specific methodology construction. In: Challenges and Strategies for Research in Systems Development, pp. 257–269 (1992)
21. Mirbel, I., Ralyté, J.: Situational method engineering: combining assembly-based and roadmap-driven approaches. Requirements Engineering 11(1), 58–78 (2006)
22. Pavòn, J., Gòmez-Sanz, J.J., Fuentes, R.: The INGENIAS methodology and tools. In: Agent Oriented Methodologies [16], ch. IX pp. 236–276
23. Ralyté, J., Deneckère, R., Rolland, C.: Towards a generic model for situational method engineering. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 95–110. Springer, Heidelberg (2003)
24. Seidita, V., Cossentino, M., Chella, A.: A proposal of process fragment definition and documentation. In: Cossentino, M., Kaisers, M., Tuyls, K., Weiss, G. (eds.) EUMAS 2011. LNCS, vol. 7541, pp. 221–237. Springer, Heidelberg (2012)
25. Seidita, V., Cossentino, M., Gaglio, S.: A repository of fragments for agent systems design. In: Proc. of the Workshop on Objects and Agents, WOA 2006 (2006)
26. Seidita, V., Cossentino, M., Hilaire, V., Gaud, N., Galland, S., Koukam, A., Gaglio, S.: The metamodel: a starting point for design processes construction. International Journal of Software Engineering and Knowledge Engineering 20(4), 575–608 (2010)
27. Slooten, K., Brinkkemper, S.: A method engineering approach to information systems development. In: Proceedings of the IFIP WG8. 1 Working Conference on Information System Development Process, pp. 167–186. North-Holland Publishing Co. (1993)
28. ter Hofstede, A.H.M., Verhoef, T.F.: On the feasibility of situational method engineering. Information Systems 22(6/7), 401–422 (1997)

Forward Self-combined Method Fragments

Noélie Bonjean, Marie-Pierre Gleizes, Christine Maurel, and Frédéric Migeon

IRIT, Université Paul Sabatier
F-31062 Toulouse cedex 9, France
Firstname.Name@irit.fr

Abstract. Developing complex systems is generally simplified if designer is guided by method from Software Engineering. However a single engineering process is often not enough to cover all the possible requirements due to different levels of expertise and systems to design. Currently, Agent Oriented Software Engineering methods aim at providing an adaptive engineering process. The method processes have been broken up into different parts called fragments, enabling the mix of different engineering processes' parts to get better adequacy between the system to be done and the process. But some difficulties still remain concerning the expertise needed to compose these fragments when the amount of fragments prevents the composition to be done by hand. This paper presents an Adaptive Multi-Agent Systems (AMAS) to deal with a new paradigm of automated fragments combining. This process is made from both the characteristics of users and system and the known fragments. Thanks to their information, agents of the AMAS self-organise and design a tailored method process. The developed system is described and then usual tests are depicted.

1 Introduction

Software reuse is generally considered as one of the most effective ways of increasing productivity and improving quality of software. To make software reuse happens, however, there is a change in the way engineers develop software: software is currently developed for reuse and with reuse. Component-based software engineering [1] is a software engineering paradigm in which applications are developed by integrating existing components. Reuse of software engineering is becoming more and more important in a variety of aspects of software engineering.

In the same way, in Agent-Oriented Software Engineering (AOSE), a lot of different methods, each with its advantages and its drawbacks [2]. Methods have to deal with these characteristics and capabilities of agents or systems. An attempt is to benefit from different methods by combining their particular features. For example, attempts have been made to combine requirements analysis in TROPOS and self-adaptation in ADELFE [3].

Coming from Situational Method Engineering, decomposing processes into pieces has interested the AOSE community because of the expected benefits of flexibility. The aim is to adapt the process to the characteristics of the business problem and to the level of expertise of engineer teams by proposing to assemble pieces of methods, named fragments, of various processes. In a first step, several teams have to split up methods into fragments and provide a precise description of them (ADELFE[4], INGENIAS [5], PASSI [6], TROPOS [7] ...). Two main results have been obtained from this step: (i) a means to precisely compare the different methods and (ii) a potential repository of fragments that will serve the community to compose new processes [8]. This kind of work is mainly done in the Foundation for Intelligent Physical Agents[1] (FIPA) context.

Currently some propositions to combine fragments have been already made, but they are mainly based on the know-how of the method engineer. In this paper, we propose a first step forward an automatic way to design a method process based on MAS technology. The process is constructed by combining fragments "on the fly" to be adapted to the specific situations of the projects at hand. The new process is based on both fragment compatibility and user characteristics. In order to respond to this need, the presented work details the use of an Adaptive Multi-Agent Systems (AMAS) which relies the cooperation of its agents to work together, making this approach especially suited to deal with highly dynamic systems such as the design of an interactive and adaptive Software Engineering Process (SEP) [4]. In this work, an AMAS is built by modelling fragments as autonomous entities.

This paper is organized as follows. First, section 2 explain the aim of this system. Then, section 3 introduces the system of Self-Combining fRagments (SCoRe) and details the behaviour of the involved agents as well as their interactions. Section 4 focuses on some usual tests and explain the results obtained. Finally, section 5 describes related works before concluding in section 6.

2 Why Such a System?

Request of Tailored Method. While the demand for specific, complex and varied system continues to grow, current methods in the MAS domain remain limited and sometimes not well adapted. For example, in order to propose a simulation-based process for the development of MASs which incorporates a simulation phase for the prototyping of the MAS being developed and for functional and non-functional validation, PASSIM was obtained by integrating method fragments from the PASSI for carrying out the analysis, design and coding phases, and the Distilled State Charts (DSC)-based simulation method for supporting the simulation phase [9]. The need for well-defined guidelines that will make the development process more efficient and more effective has become crucial. Currently, there is no single methodology that can be uniquely pointed as "the best". Until now methodology adjustments to the specific requirements and constraints are mixed in "local" adaptations and modifications. In order to succeed in creating good situational methodologies, i.e., methodologies that best fit given situations, fragment representation and cataloguing are very important

[1] `http://www.fipa.org`

activities. In particular, the fragments (sometimes addressed as process fragments, method fragments or chunks) have to be represented in a uniform way that includes all the necessary information that may influence their retrieval and assembling.

Fragment Standardisation. Method fragments are first identified by examining existing methods. These method fragments are made according to templates defined by repository designers. Therefore the choice of fragments granularity relies on designers. According to the RUP [10], the methods are defined following different levels of granularity: phase, activity and step. The granularity issue of these method fragments poses important challenges. The "step" level involves a specific and fiddly task but also requires perfect knowledge of methods and long work. This fragmentation is very fine-grained and provides a greater number of fragments. For instance, in ADELFE, the analysis phase is composed of four activities, the first of which is *Analysis of domain. Analysis of domain* consists of two steps: *Identify the active and passive entities* and *Study interactions between the entities.* These steps are related and interdependent. This low level of granularity is therefore useless and inaccurate in this situation. On the other hand, the "phase" level of granularity could form huge complete fragments. The coarse-grained granularity promotes the redundancy issue. The duplication of activities or steps may occur with high granularity. An activity or step may be included in different fragments. The risk that happens grows up with the level of granularity. In addition, the joining possibilities are therefore minimized.

Amount of Fragments. Currently, ten AOSE methods are fragmented, each one composed of approximately twenty fragments. Such fragments constitute the root constructs of the methodology itself and they have been extracted by considering a precise granularity criterion: each group of activities (composing the fragment) should significantly contribute to the production/refinement of one of the main artefacts of the methodology (for instance, a diagram or a set of diagrams of the same type). Following this assumption, fragments obtained from different methodologies are based on a similar level of granularity.

Besides, to design a process manually means studying for the compatibility of each fragment with the others i.e. approximately twenty thousand possible combinations. Although this number can be decreased by the knowledge and the know-how of process engineers, the work remains long and irksome. It is why we propose the automated combining of fragments.

Assist Designer. In our approach, a new complete process is firstly self-designed contingent on situation. The complete process enables engineers to visualise all activities and to have a whole view of the process. Then, we focus on adaptation during process execution. In every step the development team is advised on its next fragment choice according to the running features. If the features evolve, this advice may therefore differ from the following fragment initially suggested.

The studied solution resides in fragments agentification in order to design an adaptive process. This choice is justified by the problem complexity which is mainly due to the huge number of fragments. Indeed, a complex system cannot currently be designed without bugs from designers. Assist the designer during the system utilization would reduce the number of bugs and make the system most suitable to the current situation. The adaptation is therefore required. As components assembling, fragments assembling needs assembling features. In our approach, a fragments assembling is based on MAS Metamodel Elements (MMME). Two fragments are therefore assembling if one produces the MMMEs required by another.

3 Combining Method Fragments with an Adaptive Multi-agent System

The general structure of the Self-Combined method fRagments system (SCoRe) proposed is described in this section, before detailing the behaviours and the interactions of the agents composing it.

3.1 General Structure of SCoRe

We consider a method process as a set of assembled method fragments which are linked through their own required or produced MMMEs. Establishing a method process consists in linking some of the fragments toward user-defined objectives and knowledge. So, the main goal of SCoRe is to suggest a tailored process. For that, SCoRe learns the context to apply on fragments, in order to sustain this evolution. SCoRe has to act without relying on a model of the processes, meaning that it is only able to take into account the users' knowledge and needs, and to observe the evolution of the running process on which MMMEs are available, in order to decide on the fragments to add. The best possible running process is therefore designing according to a situation and the best adapted fragment has to be found at any moment.

SCoRe is composed of four distinct kinds of agents following a perception-decision-action lifecycle which cooperate according to the AMAS theory described in [11]. The basic idea underlying this cooperation consists, for every agent in an AMAS, in always trying to help the agent which encounters the most critical situation from its own point of view. Figure 1 gives the structure of a SCoRe system designing a method process. The different types of agents involved are shown, as well as the links modelling the existing interactions between them. Actually, our system is made up of:

- **MAS Metamodel Element (MMME):** required or produced by a Running Fragment, its aim is to decide fragments whom it will be linked.
- **Waiting Fragment (WF):** its purpose is to be integrate in a process once it is in an adequate situation.
- **Running Fragment (RF):** it aims at finding its place inside the running process.

- **Context (C)**: related to a fragment, it aims at evaluate its pertinence according to the MMMEs already involved in the running process and the users' needs and knowledge.

Next sections will provide a more in-depth description of these agents and interactions.

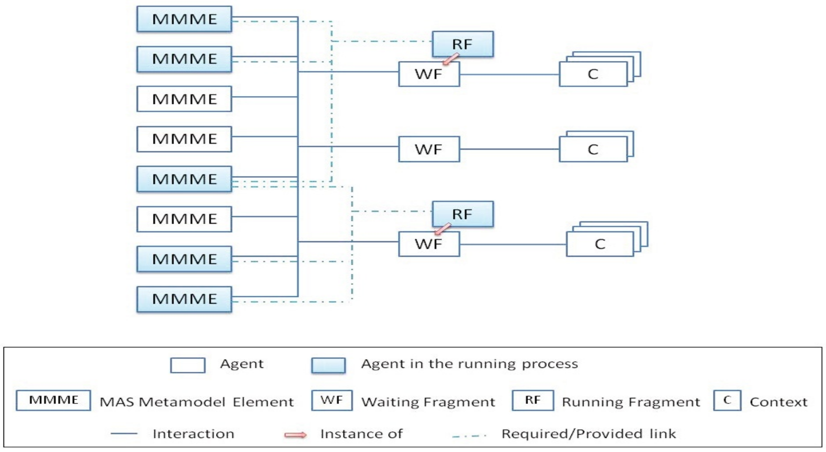


Fig. 1. Example of agents and their relationships in SCoRe

3.2 Behaviour of Agents

MMME Agents. The MMME agents represent the links between the running fragment agents. Their goal is to be incorporated in the running process. The MMMEs behaviour is represented by an automaton with two states: *non incorporated* and *incorporated*. The *non incorporated* state corresponds to a MMME linked to at least one running fragment which produces or consumes it. In this state, it requests fragments (consumer or producer). It is looking for a fragment at which it can be linked. It receives some answers from fragments with their relevancy. The most relevant fragment will be single out by the MMME agent for being put it forward as a candidate to be added in the running process. Furthermore, these agents are able to evaluate their own criticality. This criticality estimates the difference between the current and expected designed process and represents the degree of satisfaction of an agent. Therefore, the MMME agents cooperate on the selection of the most relevant fragment among the ones suggested according to their own criticality.

The *incorporated* state is reached when the MMME agent is linked with at least two fragments: one consumer and one producer. The given or required MMMEs by the designer have only to be linked respectively to at least one consumer and one producer.

Waiting Fragment Agents. The waiting fragment agents are reactive agents. Actually their goal is to notify the other agents of any requests from MMMEs. They receive messages from MMMEs which are looking for a fragment. If the waiting fragment agent considers himself as a potential solution then it forward the request to his context agents. It waits the answer from their context agent and answers his relevancy to the MMME. Should the opposite occur, the waiting fragment agent sends an answer to MMME with no relevance. Besides, when the waiting fragment agent receive a message from the MMME to inform it that it is selected, it transmits the information to the context agents. Then the waiting fragment agent spreads to create a running fragment agent.

Running Fragment Agents. The running fragment agent is created by the waiting fragment agent which represents in the running process. It is introduced on time in the process. His aim is to be incorporated in the method process. His behaviour changes according to his current state and his perception. The current state of a running fragment agent corresponds to *non incorporated* and *incorporated*. Actually, a running fragment agent is said *incorporated* when all the required MMMEs are in the *incorporated* state and at least one of the provided MMMEs is *incorporated*. Otherwise his state is *non incorporated* and the running fragment agent makes links with each MMME agent existing in the running process on which a link is physically possible. If some MMME agents are missing in the running process, the running fragment agent adds them. Furthermore, these agents are able to evaluate their own criticality. This criticality estimates the difference between the current and expected method process. It is calculated from the criticality of required or produced MMME(s) and their current state.

Context Agents. The context agents have the most complex behaviour in the SCoRe system. Their goal is to represent a situation leading to a specific method process. They do not aim to model what is happening inside the system, but rather aim at selecting the fragment to add in the current situation to reach the objectives. When such an agent finds itself in its triggering situations, it notifies the waiting fragment agent, by submitting its confidence according to its own knowledge.

In order to know when the fragment is relevant, a context agent relies on two different sets of information. First, a collection of input values represents the set of user and system characteristics. This element enables the context agent to know if it has to be triggered or not. Then, a context agent possesses a set of forecasts, which describes the impact of the action proposed on the criticality of the different variables of the system. Then, a context agent possesses a set of metrics, which describes the impact of the action proposed on the running process [12]. Those input values are modified during the life of a context agent. According to its behaviour, a context agent therefore adjusts its confidence from different feedback that it receives.

Finally, the behaviour of a context agent is represented by an automaton. Each state relates its current role in the MAS. A total of three different states exist: *disabled*, *enabled* and *selected*. The context agent can switch from a state to another thanks to the messages it receives from other agents in the system. A disabled context agent considers itself non-relevant in this specific situation. An enabled context agent thinks that it is relevant and potentially deserves to be selected. It then computes its confidence and sends them to the corresponding waiting fragment agent. Finally, a selected context agent is validated by a waiting fragment agent and its associated fragment is added in the running process. This selected context agent has then to observe the consequences of its action in order to reinforce or update its confidence.

4 Usual Tests

Considering the large number of existing method fragments, the volume of supporting studies and the users' profile, the need arises for a designed method. The designed method is conceived of not as a single interdependent entity but as a set of disparate fragments. Therefore, in order to show the rightness of a new method process, the method process has to be evaluated by several engineers for some specific system. The experience results of empirical studies that have been conducted by many practitioners and researchers. This kind of experience is complex and can take a long time to obtain sufficient results. Therefore, we firstly focus on the functional adequacy and the dynamic adaptation to specific situation.

We defined test sets corresponding on the one hand to the feasibility of this system and on the other hand to the specific situations encountered and solved by cooperation between agents. The first test is based on a set of fragments from current methods such as ADELFE[2], INGENIAS[3] and PASSI[4]. Fragments description can be found respectively on corresponding research team site. This first test aims at verifying that the system self-designs and proposes a complete method process. The sets of fragments from ADELFE, INGENIAS and PASSI enables to show the accurate behaviour of agents and their right assembling. In this case, at the set-up, all fragments are provided without order and the process is built up again. According to the users and system characteristics, one of them is therefore built up and suggested to the designer. The system is therefore able to propose the known processes.

The following test set uses fictive fragments to highlight accurate configurations. Two processes named A and B are defined. A is broken in four fragments a1, a2, a3, a4 where all fragments are sequential except for a2 and a3 which are alternative. The process B is broken in four sequential fragments b1, b2, b3 and b4. Moreover, the two processes are totally disjointed (except for last test case). They do not share MMMEs. For these tests, we chose to provide few fragments

[2] ftp://ftp.irit.fr/IRIT/SMAC/DOCUMENTS/RAPPORTS/

[3] http://grasia.fdi.ucm.es/main/node/241

[4] http://www.pa.icar.cnr.it/cossentino/FIPAmeth/docs/

Fig. 2. Test Cases: Context Adaptation (left); Dynamic Adaptation (middle); Processes Combination (right)

for better readability and to show the system adaptation by cooperation between agents which are in particular situations. Moreover, in the following tests, the both processes are shown but the most accurate method process is only designed.

The second test verifies adaptation according to users and system characteristics. Indeed, in any situation, it controls that the system advises the most adapted process. In this case, the waiting fragment agents represent only the fragments from the independent fictive processes A and B. As a result, one is chosen as the most adapted for the specified situation. This test is showed in the figure 2 (left). The double borderline around a fragment shows the process A as being the most accurate process.

The third test is about dynamic adaptation. Considering open systems, waiting fragment agents are added or removed in system during runtime. The system has to take into account these modifications and reorganises itself according to its new state. The initial conditions are the same as in the previous test. A new waiting fragment agent named b3a is then introduced in the system. This fragment more accurate for the situation is an alternative fragment to b3. It is therefore included in the running process and a new process is defined. Figure 2 (middle) shows system adaptation after the introduction of a new fragment.

The last test case shows combining processes. In this case, fragments from different processes are assembled in order to obtain a new process more adapted. In this test case, we supposed that the provided fragments from A and B are compatible. The required MMMEs are also provided by a fragment from another process. The system is therefore able to produce a new process based on fragments from both initial processes in addition to processes already known. Figure 2 (right) shows this test where the new process composed of a1, b2, a3 and b4 is advised as the most accurate.

5 Related Works

In the MAS community, the first works on fragments, their definition and their composition have been started by the working group "Methodology Technical Committee" in 2003 [8]. Currently, the working group named "Design Process Documentation and Fragmentation Working Group" aims at providing IEEE FIPA specification of fragments. The working group approach is based on SPEM extension of OMG [13], and tailored to needs of agents and MAS.

The objective of SME approach in agent oriented engineering field is to propose the most accurate process in development context. The PRoDe (PRocess for the Design of Design PRocesses) [14] approach proposes to use the MAS metamodel as a central element for selecting and assembling fragments. PRoDe contains three phases: process analysis, process design and process deployment. The analysis phase elicits requirements and leads to MAS metamodel definition. The design phase helps designer to select fragments to assemble in a new process. Finally in the process deployment phase, the new process is used to solve a specific problem.

Based on PRoDe approch, MEnSA[5] project dissents to it from analysis phase. Indeed, in this phase, requirements are used to chose fragments and fragments contribute to metamodel definition [15]. The fragments repository includes fragments from the following agent oriented methods: PASSI, GAIA, TROPOS and SODA.

The OPEN framework [16] is object oriented method based on reusable methods components. It was extend to take into account agent oriented methods and come to FAME (Framework for Agent-oriented Method) conception. FAME is an agent oriented methods repository containing for example GAIA, TROPOS or PROMOTHEUS [17].

Tools are also developed in order to make easier the methods designing by combination of fragments, as MetaMeth [18]. It is a computer-aided process engineering tool (CAPE tool) and plug-ins to assist designer during process design from available fragments included data base.

As presented approaches, ours is based on current data base of fragments and on MAS metamodel. On the other hand, it is original because it proposes an automation of fragments composition. The designer is less called upon than ProDe or MEnSA approach because the most accurate fragments are presented to him already placed in the process. Moreover, in running development, our approach can take into account process adaptation according to development context.

6 Conclusion and Future Works

This paper presented an Adaptive Multi-Agent System to design a tailored process by linking fragments together. Each agent composing the AMAS follows a local and cooperative behaviour, driven by the use of their confidence. The four

[5] `http://apice.unibo.it/xwiki/bin/view/MEnSA/`

different kinds of agents composing SCoRe were designed in order to self-design a tailored method process without relying on the method engineer. The resulting behaviour of SCoRe is the ability to design a process and adjust the proposed process according to the characteristics of application domain and users profile. This first prototype allowed to enhance our experience about practical problems such as metamodel compatibility, parameters composition or fragments adaptation to specific field.

However, there is still room from improvements in some aspects of this approach. For example, the inter-operability and the semantic matching of fragments from different methods are still missing. In this problem, some works axis on standardisation of fragments notion and of their description. The metamodel definition or ontologies for software process could be used. Another approach from model-driven engineering is the Model Transformation By Example (TTBE). The concept is to make easier model transformation writing without generic model in favour of requested generated transformation. Thus fragments drawing on similar metamodels could be made up automatically.

Moreover, another important point is the evaluation of the designed process. Actually, despite the proposal of elaborate tailored method processes, methods are built intuitively by adopting some fragments from different methods. It is therefore difficult to evaluate and compare methods. In order to made a right choice, it is necessary to evaluate the method.

Finally, this SCoRe system will be confronted to real users' problems with known method fragments, in order to allow its comparison with existing methods.

References

[1] CBSE: Component-based software engineering, 13th international symposium, Prague, Czech Republic (June 2010)

[2] Bergenti, F., Gleizes, M., Zambonelli, F.: Methodologies and Software Engineering for Agent Systems: The Agent-oriented Software Engineering Handbook. Kluwer Academic Pub. (2004)

[3] Morandini, M., Migeon, F., Gleizes, M.P., Maurel, C., Penserini, L., Perini, A.: A goal-oriented approach for modelling self-organising MAS. In: Aldewereld, H., Dignum, V., Picard, G. (eds.) ESAW 2009. LNCS, vol. 5881, pp. 33–48. Springer, Heidelberg (2009)

[4] Bernon, C., Camps, V., Gleizes, M.P., Picard, G.: Engineering Adaptive Multi-Agent Systems: The ADELFE Methodology. In: Henderson-Sellers, B., Giorgini, P. (eds.) Agent-Oriented Methodologies, pp. 172–202. Idea Group Pub, NY (2005) ISBN 1-59140-581-5

[5] Pavòn, J., Gòmez-Sanz, J.J., Fuentes, R.: The INGENIAS methodology and tools. In: Agent Oriented Methodologies, pp. 236–276.

[6] Cossentino, M.: From requirements to code with the PASSI methodology. In: Agent Oriented Methodologies, pp. 79–106.

[7] Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: Tropos: An agent-oriented software development methodology. Autonomous Agent and Multi-Agent Systems 8(3), 203–236 (2004)

[8] Cossentino, M., Gaglio, S., Garro, A., Seidita, V.: Method fragments for agent design methodologies: from standardisation to research. International Journal of Agent Oriented Software Engineering 1(1), 91–121 (2007)
[9] Cossentino, M., Fortino, G., Garro, A., Mascillaro, S., Russo, W.: A simulation-based process for the development of multi-agent systems. International Journal on Agent Oriented Software Engineering, IJAOSE (2008)
[10] Jacobson, I., Booch, G., Rumbaugh, J.: The unified software development process. Addison-Wesley Longman Publishing Co., Inc., Boston (1999)
[11] Capera, D., Georg, J.P., Gleizes, M.P., Glize, P.: The AMAS Theory for Complex Problem Solving Based on Self-organizing Cooperative Agents. In: International Workshop on Theory And Practice of Open Computational Systems (TAPOCS at IEEE 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2003) (TAPOCS), Linz, Austria, June 9-11, pp. 389–394. IEEE Computer Society (2003), http://www.computer.org
[12] Bonjean, N., Chella, A., Cossentino, M., Gleizes, M.P., Migeon, F., Seidita, V.: Metamodel-Based Metrics for Agent-Oriented Methodologies (regular paper). In: International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), June 4-6, Valencia (2012)
[13] OMG: Software Process Engineering Metamodel. Version 2.0. Object Management Group (March 2007)
[14] Seidita, V., Cossentino, M., Galland, S., Gaud, N., Hilaire, V., Koukam, A., Gaglio, S.: The metamodel: a starting point for design processes construction. International Journal of Software Engineering and Knowledge Engineering 20(4), 575–608 (2010)
[15] Puviani, M., Cossentino, M., Cabri, G., Molesini, A.: Building an agent methodology from fragments: the mensa experience. In: SAC, pp. 920–927 (2010)
[16] Firesmith, D., Henderson-Sellers, B.: The OPEN Process Framework: An Introduction. Addison-Wesley (2002)
[17] Henderson-Sellers, B.: Evaluating the feasibility of method engineering for the creation of agent-oriented methodologies. In: Pěchouček, M., Petta, P., Varga, L.Z. (eds.) CEEMAS 2005. LNCS (LNAI), vol. 3690, pp. 142–152. Springer, Heidelberg (2005)
[18] Cossentino, M., Sabatucci, L., Seidita, V.: A collaborative tool for designing and enacting design processes. In: Shin, S.Y., Ossowski, S., Menezes, R., Viroli, M. (eds.) 24th Annual ACM Symposium on Applied Computing (SAC 2009), vol. 2, pp. 715–721. ACM, Honolulu (December 8, 2009)

"Engineering" Agent-Based Simulation Models?

Franziska Klügl

School of Science and Technology
Örebro University, Örebro, Sweden
franziska.klugl@oru.se

Abstract. Multiagent simulation emerges to be one of the "killer applications" of multiagent system technology. For several reasons, there is a serious lack of engineering approaches in developing simulation models, so connecting AOSE with Multiagent Simulation seems to end in a win-win situation. A basic prerequisite is hereby to understand the current state and challenges of developing multiagent simulations. This is the objective of this contribution.

1 Introduction

Not just with the growing interest in multiagent simulation applied in domains beyond social science, the systematic development of high quality models for various objectives got into the focus of research. Multiagent simulation is a not so new simulation paradigm in which the basic metaphor of the model is the concept of a multiagent system. Active entities in the original system are mapped to agents in the simulated system. A multiagent model consists not just of agents, but also has an explicitly simulated environment and explicit treatment of simulated time. The environment can take different forms: often it forms an abstraction of a spatial environment. It may be also populated by other entities than the agents that co-exist and interact with the them.

There is a basic dilemma in multiagent simulation: on one side it is perceived as very intuitive because of the ontological correspondence between the observable, original actors and the active agents in the simulation [15]. Structure and dynamics of models can be easily understood and (informally) explained: The abstraction step between original system and simulated system is smaller compared to other simulation paradigms in which for example a group of persons is reduced to a density value. In multiagent simulation the original dynamics are reduced to activities and interactions of lower level units that explicitly generate the overall dynamics. In other paradigms, dynamics are describing and advancing during simulation e.g. based on formula that have been taken from a library and calibrated using data.

On the other side it is meanwhile common knowledge that the actual development of a useful simulation model is everything but trivial. This refers to all phases that might be identified for a development process [36]: Analysis and model design, model implementation, validation, etc. Too often, development is affected by adhoc approaches instead of a principled proceeding. The question is why this is the case? Especially, if considering the accessibility of the models and the available tools that support implementation of multiagent simulations. Thus, the objective of this contribution is to give a review of methodologies and approaches for developing multiagent simulations together with an analysis of the problematic issues.

The remainder of the contribution is structured as follows: First we describe the core issues that we identified for developing multiagent simulations as experience from more than 15 years of modelling self or supporting others. This is followed by a review of literature on engineering approaches to multiagent simulation model development. We then discuss considerations what would be useful together with an analysis why this has not been yet accomplished. The contribution ends with a short summary of the conclusions.

2 Issues in Multiagent Simulation Development

Multiagent simulation as a generic paradigm is extremely successful. N. Gilbert [15] even stated that meanwhile most microscopic simulation in social science is done in an agent-based way. Several introductory text books have been recently published ([35],[15], [40] to name just the more general ones. As indicated in the introduction, there are mainly two reasons for this triumph: the intuitiveness and the flexibility of the paradigm. The ontological correspondence between agents and original actors facilitates understanding of the model. The unit of description is the active entity in the model: Real pedestrians are mapped to agents in crowd simulation or households in demographic models. The second major advantage is the freedom of design. Heterogeneous entities, heterogeneous spatial environments, arbitrary complex agent decision making ranging from so-called “zero-intelligence” agents (as in [38]) to agents using complex cognitive architectures (such as SOAR [47]) can be found in multiagent simulation models. In other simulation paradigms, heterogeneity of entities or space, structural variation (entities leaving or entering the system) or context-dependent and flexible individual decision making are hard to achieve in a direct way [29].

2.1 Understanding and Design

Considering the different system/model representations in the general phases of development[1], one can identify why and where issues occur. Figure 1 illustrates these phases starting from the “original system”, the system that the modelling and simulation endeavour is about. The starting point for the modelling process is the representation that the modeller[2] has about the original system. This forms some kind of hypothesis about how the original system behaves or works. As it represents how the modeller understands the original system, it is framed already by the tools that the modeller uses to analyse the original system and thus by the experience of the modeller. Starting from this rather vague system hypothesis representation, a model concept is developed that relates elements of the original system – as seen through the glasses of the analysed picture – to elements in the model. These are the conceptual design activities in which

[1] We are ignoring for now other phases of a full simulation study as given for example in [31].

[2] In [6] different participants in a multiagent simulation study are introduced, see below. We just use the “modeller” here for all responsible persons, either resembling the modelling group or the person who unites all capabilities.

the modeller decides about the level of detail of the model; here the modeller formulates, abstracts or concretises, selects the elements that form the core of the model. This conceptual model is then further substantiated and implemented for generating the executable model that is used for simulation runs.

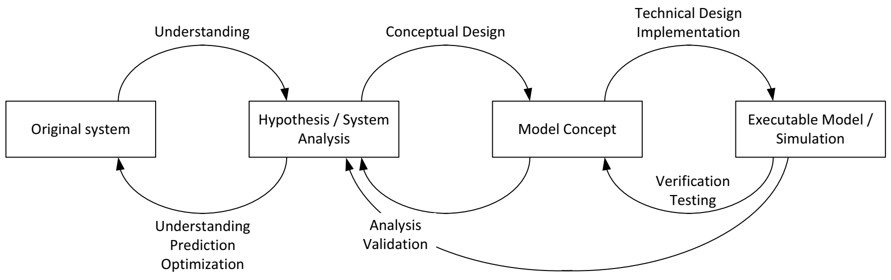


Fig. 1. Different system representations in the early phases of a simulation study in general

In each of these transitions, there are particular issues. For multiagent simulation in general the glasses with which to look at the original system provide a high ontological correspondence between the original system – if there is something like situated actors as in many scientific or complex industrial domains – and model concept compared to other modelling paradigms. For example for a System Dynamics model the modeller needs to has to see the flows between variables or for a Queueing System, the metaphor with which the system is looked at consists of queues and servers – that means a bigger abstraction need for the basic system understanding. So for the design activities, we assume that a form of understanding of the system containing a multiagent system as a central part is given. Issues start with the design steps in which a rather vague understanding of the original system needs to be pinned down to more concrete model representations. Other metaphors require a bigger step in understanding, but can be more directly formulated in a more precise way. In the following we shortly discuss the most important issues.

2.2 Micro-macro Link

A basic reason for its attractiveness for particular domains, is the generative nature of the multiagent simulation paradigm. The structure and dynamics of the overall system are not directly described, but generated from behaviour and interactions of simulated, individual situated actors (see also [8] introducing the multiagent simulation approach as "generative social science"). So, there are at least two levels of modelling and observation: the low-level agents and the aggregate system level. "Running" the low level produces the structure and behaviour on the aggregate level; in case of models involving institutional effects, the macro level may be explicitly defined in addition and influence the agent-level behaviour. In general, a formal priori analysis before simulating the system is hardly possible, only by running the simulation, the what, where and when of a social phenomenon "emerging" from the low-level agents, can be fully determined.

In many applications a certain macro level phenomenon in the original system is to be reproduced or optimized. As it is generated by the interaction of micro-level behaviours, the micro-level rules[3] determining the behaviour have to be adapted in a way that the intended aggregate phenomenon is produced. As, the connection between both is not always clear or may even not exist, the intended macro level phenomenon cannot be used to systematically derive the appropriate agent rules. Whether it works or not, can only be tested via simulation. This makes the adaptation of the lower level behaviour bound to experience-based exploratory procedures and to a somewhat arbitrary try and error procedure (as already discussed in [44]). This is insofar problematic, as the development of multiagent simulations as computational approach should be performed with sufficient rigour to generate useful and reliable models. Exploratory, experience-based procedures can hardly be coined as principled. Such a procedure can also integrate documentation and advanced testing approaches, yet the basic question on how to come up with the appropriate low-level behaviour is left to individual creativity and experience.

2.3 Level of Detail/Abstraction

As stated above, the multiagent paradigm imposes only little constraints beyond the basic conceptualization. Yet, such constraints could serve as a kind of guideline how the model can be substantiated during model design. A modeller is left with the idea that *everything* can be included in the agent behaviour or as environmental process. It is not a priori clear whether to include a simulation of rain cleaning the surface or just a random process that from time to time deletes all pheromone trails at once. There is a temptation to formulate behaviour that the modeller thinks is "reasonable", ignoring that every aspect of decision making and behaviour is an assumption that needs to be justified. If there is a formally captured underlying theory (as e.g. in Physics), the question which processes to include can be clearly decided on. Unfortunately, the existence of such a theory is not given in many application domains for multiagent simulation. Although following the value of "Keep It Simple" is clearly comprehensible in theory, it is not easily implementable: It often remains unclear, what is the most simple model. The modeller may be trapped between interpreting something that is produced by hardly more than random behaviour, or on the other extreme is adding unnecessary complexities to the overall model using complex cognitive models for actually simple decision making. [7] argued against over-simplified models, but demanded comprehensiveness for all levels of agent behaviour. This forms a good starting point for an iterative procedure involving model abstraction techniques [10]. Deciding on the level of abstraction means deciding on what entities are explicitly populating the simulated world and what processes govern, what information is used in the decision making of the agents. Replacing complex processes with a random distribution is a standard abstraction technique. Describing decision making behaviour in a form of rules may replace the use of complex cognitive processes. Simple state automata may represent a

[3] The word "rules" is used as a placeholder for all kinds of agent-level programs determining the action of the agent from its perception and internal beliefs, plan schemata,... There is no restriction on the underlying architecture. Flexible elements such as tune-able parameters, optimize-able plan schemata, automata, etc. can be found in any architecture.

wealth of lower level metabolistic processes. There are many abstraction techniques for simplifying model while mostly maintaining the validity of the model .

2.4 Critical Parameter Structures

Depending on the complexity of the multiagent simulation model, a huge number of parameters need to be filled with appropriate values[4]. These parameters may be factors in formulas or thresholds for decision making. Basically everything with absolute value in the model can be seen as a parameter. Also the initial values for state variables of all entities are parameters. Sometimes, the model is intentionally designed to have many parameters for serving as some form of information storage for a huge amount of data, yet in principle the more parameter, the more difficult the justification of assumptions and the calibration becomes.

Not only the sheer number of parameters may be critical, but also the nature of the parameter themselves. A single parameter in the agent behaviour can have an enormous effect on the overall aggregated behaviour. Practically, the modeller just sets one value in the behaviour description that is then instantiated to many agents using that value. Effects are even worse, if there are non-linear feedback loops or if the parameter is not just regulating some process such as energy consumption during a particular agent activity, but serves as a decision threshold. Izquierdo and Polhill [24] call them "knife-edge parameter", if the behaviour of the agent changes depending on this parameter. An example is, if the energy level of the agent is higher than a threshold, the agent starts to reproduce; below the threshold, it just performs some random search for food. Setting such a parameter homogeneously for all agents may cause dramatic effects such as a whole population exhibiting the same behaviour change within short time. Even if there are not so obvious artefacts, such threshold parameter can cause chaotic behaviour: small changes in the parameter values result in completely different phenomena. Simulations with such parameters are highly sensitive, and thus hard to calibrate. This is also the case, if parameter values are not independent from each other.

Thus, agent-based models are often very sensitive to parameter changes. It is problematic for model stability and also for model reliability if the discussed outcome of the model can be only produced with a small range of parameter values. The problem worsens because a high level of detail comes also with a high number of parameter.

Besides the problem of many parameters and complex calibration, there is a much more essential issue related to parameters: If there are enough of them, a model cannot be falsified any more. That means, that a modeller cannot show that the model is *not* corresponding to the original system, as with sufficient parameters, there can be always a combination that produces any overall, aggregate behaviour. This is critical as such a model is basically useless if the objective of the simulation study is to better understand the original system by generating hypothesis about which or whether at a there is low level agent behaviour that can produce the observed overall output.

[4] This process of determining the set of parameter values for producing the most valid behaviour of model is called calibration. In general, it has clear similarities to model optimization [9]

2.5 Technical Aspects

Besides those principled problems, there are also hardly surprising challenges in the technical design and implementation. Understanding of the modelled system may be increased by the modelling procedure alone, yet, if an analysis of the dynamics of the system is to be made, implementation is necessary. This is challenging in a way similar to multiagent systems. Simulated multiagent systems are also consisting of distributed intelligent decision makers, each with its own thread of control, its local beliefs and interacting and acting in parallel. In addition to these aspects that also form practical challenges when programming of multiagent systems, some additional particularities have to be considered.

- There are extended design choices about the environmental model as well, as the simulated environment is fully[5] controlled by the modeller. That means, for facilitating the design of the agents, the environmental model can be adapted. Many simulations are using these ideas. A prominent example are crowd simulations using grid environments for carrying gradient field data or navigation graphs supporting the path finding of the agents. Thus, information is explicitly stored in the environment that does not have a correspondence in the original system, but is just put there to facilitate the agent implementation. This is acceptable depending on the objective of the simulation. Questions, of how and where information that the agents need for decision making is represented are not so easy to decide upon, efficient solutions might be preferable to solutions that reproduce complex perception processes on a one-to-one basis.
- During simulation, virtual time is advanced to express the dynamics of the model. As environment and time are artificial, the modeller needs a way for explicitly handling artificial parallelism of the agent's update. In principle, every agent could run in its own software process, but for simple agents introducing artificial parallelism is a more efficient option. There are several ways to handle virtual parallelism. Depending on used infrastructure, the modeller has to take care about these low-level aspects of simulation implementation and has to fully understand the con sequences of his choice.
- As a consequence of artificial shared time and environment, there are additional ways of coordination and synchronization of agent activities such as implicit coordination based on stigmergic interaction. Similar concepts were introduced to the multiagent software community, the most prominent is the Agents & Artefacts (A&A) framework [42]. This increased richness of expressible, explicit and implicit coordination may also form a challenge for modellers in case the original coordination cannot be directly mapped to the simulation or the approach just tackles an abstract form of interaction on the conceptual level.
- Scalability also forms an issue on the technical level. Multiagent simulations need to be run with appropriate numbers of agents. If the development of the load of a highway network is to be shown over time, only a small number of agents is not sufficient (see [41] for a simulation of complete Switzerland with several million

[5] At least in all simulation without a participatory component, that means without involving interaction between agents and real humans.

non-trivial agents). To implement such models forms a challenge for a modeller, especially if there is no supercomputer hardware accessible. Issues of conceptual scalability have been discussed above.

Considering these issues, one can understand why developing a multiagent simulation model is not trivial, neither for persons with training in abstract conceptualization and the usage of programming languages, nor for domain experts that deeply understand the original systems. If multiagent simulation shall be sustainably successful, principle-guided development is essential. Engineering approaches are inevitable for reliable and useful models and lift teaching multiagent simulation from simple training in using a simulation platform. Tackling similar problems should lead to similar models independent from the particular modeller. In the following section, we will give a short overview over existing suggestions for supporting the modelling and model implementation activities.

3 Engineering Approaches to Multiagent Simulation

The failure of producing equivalent results from models tackling similar phenomena [2] indicated quite early that there is a problem in principled development of multiagent simulations. Since then, many suggestions have been made to tackle this problem and introduce systematic development into multiagent simulation model construction and usage. Yet, one can still read statements such as: "Current practice is that most ABMs are developed from scratch and that the choice of model structure and process representation is more or less *ad hoc*" ([19], p. 362, Emphasis in the original). Nevertheless, there are a number of works that aim at improving this situation. We categorized them into a) general guidelines often developed from standard simulation engineering, b) works proposing adapted methodologies that aim at giving further, specific support for the design of multiagent simulations.

In our analysis, we ignore that there are hundreds of platforms and tools for multiagent simulation. A wikipedia page[6] lists 78 tools that can be used for multiagent simulation. Heath et al. [22] show in their survey that the variety of tools used in published multiagent simulation is huge: in the 279 publications that they included in their survey, the usage of 68 different tools was listed – although about a third did not state what platform they used for implementation. Considering introductory textbooks and announced course, the variety of tools introduced is much more restricted. Railsback and Grimm [40] exclusively use NetLogo[7]; also a course in Spring 2013 at the Centre for Policy Modelling at Manchester Metropolitan University was using NetLogo[8]. There are many more courses given by the developers of platforms, such as Repast[9].

As platforms and software for multiagent simulation have been already surveyed sufficiently, we focus on methodologies, clearly the meta-models underlying those platforms

[6] `http://en.wikipedia.org/wiki/Comparison_of_agent-based_modeling_software`, accessed at 2013-04-12

[7] `ccl.northwestern.edu/netlogo/`, accessed at 2013-04-12

[8] `http://cfpm.org/simulationcourse/`, accessed at 2013-04-12

[9] `http://repast.sourceforge.net`, see `http://www.dis.anl.gov/conferences/abms/info.html` (accessed 2013-04-12) for a course in May 2013

are helpful to convey how a model should be basically structured, yet they are much to implementation-specific to help with the most urgent issues given above. Thus, in the following we will give a short review of works in the two above mentioned categories.

3.1 General Guidelines and Best Practices

Most literature aiming at guiding modellers and introducing systematic approaches to multiagent simulation suggests processes similar to what can be found in general, introductory simulation literature such as [31]. A simulation study usually starts with problem formulation, fixing the objective of the simulation study, this is followed by system analysis and data gathering activities. This results in a conceptual model or model design that is then implemented using appropriate infrastructure. Pilot runs and sensitivity analysis are used to test the simulation, experiments are designed and performed, data is analysed and the results documented, presented and used. Validation (testing whether the right model is used) and verification (testing whether the translation between the different phases is correct) are mandatory activities throughout the simulation study. Such a sequence of activities – with more or less explicit conceptual modelling phases – can be found in basically all general texts on simulation. This sequence of activities forms also the basis for works aiming at engineering multiagent simulation:

Richiardi et al. [43] suggest a methodological protocol to follow with those activities explained in the context of social science agent-based simulation. Drogoul et al. [6] also give a generic process model. For that, they identify different roles that participants may play in that study. There is the "Thematician" who is basically the stakeholder who is responsible on the domain and problem side, the "Modeller" who does a careful specification of the model and the "computer scientist" who basically implements. Starting from these different roles and thus expertise, Galan et al. [12] analyse what errors and artefacts may be generated in the different activities and when communicating between each other. They define errors as "mismatches between what the developer believes a model is and what the model actually is" (paragraph 4.2 in [12]), artefacts relate to assumptions with significant effects which are considered as not important. Based on that analysis Galan et al. develop some guidelines embedded into a process model showing how to avoid these errors. These activities mostly consist of advanced testing and reformulating and re-implementing (parts of) the model using other languages and tools – which is clearly a useful experience but is often practically impossible in restricted project times. Yilmaz and Oren [48] connect agent-based simulation with relevant systems engineering approaches, unfortunately staying on a very general level. North and Macal [35] is an introduction on agent-based simulation from a practical point of view. They give extensive background on the concepts behind agents and multiagent systems, but methodological guidelines are restricted to information on toolkits and simulation infrastructure, verification and validation. Descriptions are extensive and on a level that is broadly understandable and based on many examples. Interestingly, they use UML diagrams for presenting examples in concise way[10]. Also Gilbert in [15] - a small, condensed

[10] Standard UML (still 1.0 then) is suggested for describing and specifying multiagent simulation models also by [37], [5] or [4]. Interestingly one cannot observe a similar tendency to extend UML as in the AOSE community.

introductory booklet - introduced to agent-based modelling focussing on particular concepts and examples. He also gives advice on verification and validation as well as on implementation infrastructure, yet in general not more formal and engineering-oriented. A more full life cycle is given in Abdou et al. [1] yet in a generic way, but enriched with examples. In their general social science simulation book, Gilbert and Troitzsch [16] include a chapter on how to develop multiagent agent models containing a mixture of general steps shown with an example. The design starts with the structures of agent types, adding dynamics and interaction to their behaviour. They shortly show how to use UML as language to describe the example. Also, Railsback and Grimm [40] embed their overall systematic way of developing and describing multiagent Simulation models in a general life cycle. Their main contribution to the development of multiagent simulation models has two particular elements: the ODD protocol for describing models and the Pattern-oriented Modelling approach. Both will be described in the next subsection.

Norling et al. [34] suggest a completely different "process model" containing basically two phases: exploration and consolidation. They argue that systematic approaches derived from software engineering are not appropriate for social science simulation. For the type of simulations they address, the objective is to understand a particular phenomenon or process by creating a model that generates it. To their view, social science modelling is mainly a creative endeavour, not an engineering one. Understanding what the modeller is actually formulating and what the model actually does is central, as grounded statements have to be derived from the model. Thus, the consolidation phase contains activities such as extensive checking or documentation. Also, for the more creative exploratory phase of their informal approach, Norling et al. give a set of best practices. They do not tackle how the creative process can be supported, but focus on how to keep control "despite of" the creative process.

Helbing and Balietti [23] elaborate on how to do agent-based simulation giving principles as a set of fairly general and informal "you should" statements, such as that assumptions should be documented. A similar list of best practices can be found in the annex of Miller and Page [33]. Examples are "Keep the model simple", "Avoid Black Boxes" or "Write good code". Yet, these are statements that in their generality are true for all forms of simulation and do not address the particularities of multiagent simulation development.

Summarizing, introductory tests and also courses (including my own at the University of Örebro) are using many examples for illustrating also the underlying concepts and how to proceed when constructing an agent-based model. Systematic life cycle or process models taken from general simulation engineering or standard software engineering processes are useful per se, as they show how to systematically end up in a documented, understood model. However, with the issues presented above in mind, would it not be better to have more specific approaches? A few specific methodological approaches have been suggested, they will be reviewed in the next section.

3.2 Specific Methodologies

Only few approaches have been proposed that were specifically adapted to the development of multiagent simulation models. Some only consider single steps of the overall life cycle, such as validation [26]. [27] summarizes different abstract approaches to model design, yet without giving any language to support the different design strategies.

As mentioned above, Railsback and Grimm made two major contributions to the development of multiagent simulations: The establishment of the ODD protocol (originally published in [17]) and the Pattern-Oriented Modelling approach (first published in [18]). The ODD protocol for documenting multiagent simulation models was developed in reaction to the frequent failure of describing models in a way that they can be replicated or fully understood. It gives a standard structure of what elements to tackle when documenting an agent-based model. The basic idea behind is to describe a model from overview to details. A central part is the characterization of a model in terms of design concepts such as Adaptation, Emergence, Interaction, etc. These design concepts also give the modeller a guideline what questions to ask when conceptualizing and designing a multiagent model. Thus the overall approach is not just helpful in documenting, but more general for conceiving models. The second contribution, the Pattern-Oriented Modelling gives an abstract guideline for some form of iterative development of simulation models in general. Starting point is a set of data patterns, such as a particular height distribution or spatial distribution of a phenomena – in social science literature these would correspond to stylized facts. These patterns are used to determine whether model development must be continued or not. Also, mainly applied in individual- and agent-based simulation, this forms a very generic procedure. A few more specific methodologies have been suggested during the last years:

MAIA ([14]) is a recent suggestion of a meta-model that supports conceptualization of a multiagent simulation based on (scientifically grounded) institutional analysis with different views on the overall system. A particular focus is put social structure involving also norms, explicit modelling of dependencies or plans. The usage of the meta-model is supported by web-based interface and tools for code generation. MAIA per se is designed to support creation of complex social simulation models, whether the provided tools actually scale with institutional and behavioural complexity needs to be shown.

IODA ([30] focusses on reactive agents. Analysis starts from the interactions of the different agent categories which are interpreted in a wider sense including also actions thus capturing also stigmergic interactions. Also treating agents and non-agent objects in the similar way, allows an overall uniform approach formulating the model design. From the specification of interactions, the behaviour description of the reactive agents can be directly derived and corresponding code is generated.

Fuentes-Fernandez et al. [11] show how the INGENIAS meta-model is apt for model-driven development of multiagent simulations. They illustrate the different steps for using the meta-model, mapping domain concepts to element of the meta-model such as agents, roles, goals, interactions or society. With the refinement of interactions, the domain expert or “thematician” has produced a model specification that then can be transformed into executable code. The meta-model does not explicitly tackle time or space, so it seems to be apt more for complex social models in which only direct interaction between agents is relevant.

A methodology that proposes a similar proceeding from system analysis based a quite traditional meta-model, subsequent model design steps that then lead directly to executable code is *easyABM* [13]. The meta-model contains particular perspectives for agents, artefacts and the societal level and is refined for code generation. Explicit treatment of time is missing, representation of space is only present on the design level. Garro and Russo [13] give container transshipment management as the example which could be also result in a working multiagent system, not a simulation model. Also other AOSE methodologies have been successfully applied to multiagent simulation. For example, the *Adelfe* approach seems to be particular apt for domains in which the conceptual model is not fully clear [3] or in which self-organization plays a particular role [20].

The MABLe methodology [25] is build on the hypothesis that it is easier for a modeller to describe what is good behaviour than to specify and implement the agent-level behaviour. The modeller gives an objective function characterizing valid behaviour, an environmental model and a specification of the agent interfaces. The agents then use agent learning techniques to generate their behaviour model which is then evaluated by the modeller.

A methodology that allows capturing complex (social) scenarios using an appropriate not too technical meta-model, enables the development of richer simulations, especially if the models need to scale with respect of behavioural and social complexity. Clearly, methodologies originating from AOSE approaches in which the organization of agents forms the central starting point from which institutions, behavioural norms, agent goals and interactions are derived, enable the development of such complex social models. Yet, they start from a structural point of view. Can they really help with the issues given above? Can they help to discriminate between agent rules that need to be included or not? What about agent behaviour that only indirectly contributes to the overall organization and in which the agent is not interacting with others? What about temporal aspects, such as durations of activities or actions? On the other hand, it is quite obvious that really specific and concrete approaches such as IODA or MABLe hardly scale to more complex models. Yet they focus on the dynamics and not so much on structures. How shall an appropriate meta-model for multiagent simulations look like? In [28], we started to develop a very basic meta-model for multiagent simulation containing no social constructs up to now. But, environmental structures are given that can capture a wide variety of spatial representations. There is also a clear distinction between agents and resources which are entities on their own right corresponding to non-active entities in the real world. As being so basic, the meta-model does not scale with increasing complexity of the agents and social structures they are embedded with. So, an essential next step is to merge the basic simulation-oriented meta-model with an appropriate AOSE meta-model.

3.3 What do We Need for "Engineering" Multiagent Simulations?

If all these guidelines and advices would be taught and used in practice, this would be a huge improvement, also on the reputation of multiagent simulation. Models would be developed with the same rigour as in other computational simulation approaches. Yet, is this sufficient? Sufficient for teaching multiagent simulation, so that a novice might be not as efficient as an experienced modeller, but the result would possess substantial quality?

For answering these questions, we have a look at what it would mean to have a full engineering approach to multiagent simulation. This leads to the basic question, do we have the knowledge to do engineering as done in other engineering domains. For that aim, we have a look at the seminal work of Vincenti [45] who analysed different cases from aeronautics for identifying what is knowledge is involved in engineering. Following the findings of [45], there is particular engineering knowledge that is distinct from scientific knowledge, showing that the engineering of an artefact is not just applying its underlying scientific principles, but needs additional skills. Vincenti identified the following categories of engineering knowledge:

– Knowledge about the fundamental design concepts
– Criteria and specifications
– Theoretical tools
– Quantitative data
– Practical considerations
– Design instrumentalities

Can we identify knowledge that fills these categories for multiagent simulation as a prerequisite for successful multiagent simulation engineering? Is this knowledge already existing or does it need to be generated? In the following we will tackle each of the categories examining the state of available knowledge.

Knowledge about Fundamental Design Concepts. There are two dimensions in which design concepts are relevant for multiagent simulation development: 1) concepts that govern the dynamics of multiagent systems and 2) concepts that help with the modelling and simulation, that guide how to develop a high-quality, useful simulation study. For both of these categories, there are suggestions: As mentioned above, [17] list a set of design concepts for multiagent simulation for structuring documentation. These concepts can be also used for guiding the design of a model: *Emergence* (which phenomena emerge, which are imposed?), *Adaptation* (how do individuals adapt for improving potential fitness?), *Fitness* (how fitness-seeking is modelled?), *Prediction* (do individuals predict their future?), *Sensing* (What information can the individuals access internally and in the environment for decision making?), *Interaction* (What kind of interactions are assumed?), *Stochasticity* (why the model has stochastic elements?), *Collectives* (are the individuals grouped to some form of collective or group?), *Observation* (what data is collected for testing, understanding, analysing the model?). These list of design concepts was updated [39] adding Basic Principles and Learning. Fitness was generalized to Objectives. Yet, these design principles have two problematic aspects: This is an incoherent list: there are concepts that tackle the internals of the model, others describe the model from outside. This should not be mixed. The second issue is that it is not clear which concepts are relevant for which type of model. The authors suggest to tackle just the relevant ones and ignore the others. One could clearly see that the original proposal was developed for ecological multiagent simulations, in which organization and social aspects are not so important. We would expect that the next revision of the ODD protocol design concepts will put more emphasis on cognitive processes of the agents, on feedback loops between the social, organizational level and the individual. For future research, a categorization of models with respect to relevant design concepts would be a good starting point advancing the knowledge on what is relevant for multiagent simulations.

The second level of design concepts refers to modelling and simulation in general. What is a good model and what activities have to be done for developing a good model? On that level, there is already a lot of knowledge coming from more technical approaches to simulation, starting with the seminal book by Zeigler [49]. Books such as [31] have accompanied generations of students and practitioners. Modelling and simulation are standard activities when it comes to plant design or technical systems. This knowledge must not be ignored, although original systems are not so well understood, reliable data is not available to a similar extend and the objective behind multiagent simulation studies is often more oriented towards theory building. All the approaches given in Section 3.1 are grounded on this established knowledge from technical simulation. However, these differences are not emphasized, general modelling process models for multiagent simulation should focus more on iterative, agile approaches as this fits more to the uncertainties on the original system. Interestingly, Willemain [46] could empirically demonstrate that experienced modellers in general do not follow the systematic design processes, but apply a more prototype-oriented, exploratory development. Specially important are validation and verification techniques. These must be developed to use informal knowledge in a systematic way instead of being mainly based on empirical data that is neither available nor easily generate-able for the typical domains of multiagent simulation. Suggestions such as [32] are just a first step to that.

Knowledge on Criteria and Specifications. What is a good multiagent simulation model? Besides standards on the technical level, there is not much to determine how good a model is, independent from the objective: A good model is the one that is sufficiently valid for satisfying the objectives of the simulation study. Validity is not just an aspect of quality, but an essential prerequisite for model usage. On a technical level, additional features characterizing a model of high quality are associated with the clarity of the model design, implementation and documentation that make the model and its results understandable to many people, maintainable and reusable. This aspects are quite obvious and rooted in general simulation engineering. Yet, how to address these general criteria for the specific case is not fully clear. Some of them relate to good software development that can be taught as the basic form of handicraft. On a design level, developing templates for particular domains or problem categories could be a starting point. In lectures on multiagent systems, students learn algorithms for task allocation or agent strategies for negotiation that are good solutions for particular problems. In multiagent simulation, the algorithms that the simulated agents use must be derived from the original system. So for central aspects of the model, only algorithms can be used if they sufficiently correspond to the original system, not if they are proven to work. Developing such model design templates could be possible for particular domains, such as ecology. The chances are high there due to long history and experience in modelling and simulation. For example for mathematical, macroscopic simulation in biology, Haefner [21] gives a library of functions with information for which type of phenomenon a formula can be used and which parameters have to be estimated from data. Agent behaviour or interaction patterns could be a starting point, if it is clear for what type of domains, for what kind of agents, addressing which design concept, they could be used.

Theoretical Tools. Theoretical tools for the type of engineering that Vincenti considered are related to mathematics toolsets or even to modelling and simulation. These theoretical tools can be used for evaluating a design or predicting its properties without actually implementing it, also for deriving a part of the design from the other settled elements. Considerations what would be the corresponding theoretical tools for multiagent simulation, could lead to current meta-modelling approaches, to modelling on different levels of abstraction with automatic or systematic connection between the levels. These tools form the basis for model-driven development of multiagent simulation models (see current developments given in Section 3.2). Other research that can contribute to the advancement of theoretic tools is related to model checking of multiagent systems. There is still a lot of research necessary to make these tools applicable for modelling problems of realistic size.

Quantitative Data. Quantitative data is basically related to constants that can be used or have to be considered during model development. This cannot be tackled in general for multiagent simulation, but only for particular domains. For some domains, known constants exist, e.g. the mean speed of a pedestrian (depending on the appropriate cultural background and activity), for others such constants need to be found. This is a problem of the underlying research area.

Practical Considerations. Practical considerations refer to heuristics, on when to use a particular approach or solution for practical reasons. An example are considerations to trade correctness against transparency in algorithms dealing with virtual parallelism. One needs to fully understand the way how models are used, what are the requirements from a practical point of view and what can be actually traded. To capture this knowledge explicitly, is hard. Mature and experienced modellers can make practical considerations, yet to our knowledge this is not formally written down. Best practices as given in [33] are guidelines for building good models. That means these are more like design principles rather than knowledge that allows reasoning with respect to the relation between design and practical usage of the models.

Design Instrumentalities. As mentioned above, there is a broad variety of platforms and tools that can be used for implementing multiagent simulations. Platforms such as NetLogo, Repast or Mason are all based on a more or less explicit meta-models which are actually quite low level. They add basic agent concepts to programming languages. Also, tools that support visual development, such as SeSAm, are not much more elaborated considering the underlying meta-models. Tools provided with recent methodologies provide a more appropriate instrument enabling more complex models. Yet, their meta-models must be derived from theories in the prevailing domain in which the platform is to be applied. The meta-model cannot be simply derived from concepts in AOSE, nor shall be adhoc. They form an important basic (hidden) assumptions for the model development. The more restrictive and the richer the underlying meta-model is, the more assumptions have to be taken and justified.

A clear identification of particular types of models and meta-models, probably separately for different domains, is important for identifying similarities and difference of needed instruments. Based on that analysis appropriate instruments for model design can be proposed, for example for models with a focus on organization and multi-level interactions, for models with discrete choice agents or for models in which agents just share a common environment.

Conclusion. In the same way as there is not a unique type of multiagent systems, multiagent simulations are used in too many domains for admitting a uniform point of view on them. One can identify many dimensions according to which one can characterize such models. One of the most important is the objective that the modeller pursues when building the model as from that requirements are put on the model. Also, the empirical embeddedness and level of abstraction are important categories, as they also define requirements and the type of data which is available for validating the model. Yet, these dimensions are on a meta level, they do not tackle the contents of the model. Here the role of the environment, the type of interactions used, etc. would be relevant. These are the aspects that are actually tackled when designing a multiagent simulation model. The issues on that level are the actually challenging ones. So, summarizing our findings, one can identify the following steps for research advancing towards engineering multiagent simulations:

1. Identify sub-categories of multiagent models with corresponding sets of different design concepts,
2. Develop a shared meta-model for multiagent simulation models generalizing over clearly identified types of models and providing a clear terminological basis. This terminology should be very clear on what are elements of the model and what belongs to the infrastructure necessary to run the model.
3. Adapt engineering approaches to the identified sub-categories and provide specific instruments, starting from the general simple meta-model.

Clearly, there is still a lot to do to create a true engineering approach to multiagent simulation model development and usage. It is even not clear whether this can be really accomplished. These three steps are just the first tasks, but with their success, we would be a big step further.

References

1. Abdou, M., Hamil, L., Gilbert, N.: Designing and building an agent-based model. In: Heppenstall, A.J., Crooks, A., See, L.M., Batty, M. (eds.) Agent-based Models in Geographical Systems, pp. 141–166. Springer (2012)
2. Axtell, R., Axelrod, R., Epstein, J.M., Cohen, M.D.: Aligning simulation models: A case study and results. Computational and Mathematical Organization Theory 1, 123–141 (1996)
3. Bernon, C., Capera, D., Mano, J.-P., Videau, S., Regis, C.: Towards Self-Modelling of Metabolic Pathways. Journal of Biological Physics and Chemistry 9(1), 43–50 (2009)
4. Bersini, H.: UML for ABM. Journal of Artificial Societies and Social Science 15(9) (2012)
5. Bommel, P., Müller, J.-P.: An introduction to UML for modelling in the human and social sciences. In: Phan, D., Amblard, F. (eds.) Agent-Based Modeling and Simulation in the Human an Social Sciences, pp. 273–294. Bardwell Press, Oxford (2007)

6. Drogoul, A., Vanbergue, D., Meurisse, T.: Multi-agent based simulation: Where are the agents? In: Sichman, J.S., Bousquet, F., Davidsson, P. (eds.) MABS 2002. LNCS (LNAI), vol. 2581, pp. 1–15. Springer, Heidelberg (2003)
7. Edmonds, B., Moss, S.: From KISS to KIDS - an 'anti-simplistic' modelling approach. In: Davidsson, P., Logan, B., Takadama, K. (eds.) MABS 2004. LNCS (LNAI), vol. 3415, pp. 130–144. Springer, Heidelberg (2005)
8. Epstein, J.M.: Generative Social Science: Studies in Agent-Based Computational Modeling. Princeton University Press (2007)
9. Fehler, M., Klügl, F., Puppe, F.: Techniques for analysis and calibration of multi-agent simulations. In: Gleizes, M.-P., Omicini, A., Zambonelli, F. (eds.) ESAW 2004. LNCS (LNAI), vol. 3451, pp. 305–321. Springer, Heidelberg (2005)
10. Frantz, F.K.: A taxonomy of model abstraction techniques. In: Proceedings of the 27th Conference on Winter Simulation, WSC 1995, pp. 1413–1420. IEEE Computer Society (1995)
11. Fuentes-Fernandez, R., Galan, J.M., Hassan, S., Lopez-Paredes, A., Pavon, J.: Application of model driven techniques for agent-based simulation. In: Advances in Practical Applications of Agents and Multiagent Systems, PAAMS 2010, Salamanca, Spain (April 2010)
12. Galan, J.M., Izquierdo, L.R., Izquierdo, S.S., Santos, J.I., del Olmo, R., Lopez-Paredes, A., Edmonds, B.: Infrastructures and tools for multaigent systems for the new generation of distributed systems. Engineering Applications of Artificial Intelligence 27(7), 1095–1097 (2011)
13. Garro, A., Russo, W.: easyABM: A domain-expert oriented methodology for agent-based modeling and simulation. Simulation Modelling Practice and Theory 18, 1453–1467 (2010)
14. Ghorbani, A., Bots, P., Dignum, V., Dijkema, G.: MAIA: a framework for developing agent-based social simulations. Journal of Artificial Societies and Social Simulation 16(2), 9 (2013)
15. Gilbert, N.: Agent-based Models. In: Quantitative Applications in Social Science. Sage Publications (2007)
16. Gilbert, N., Troitzsch, K.G.: Simulation for the social scientist, 2nd edn. Open University Press (2005)
17. Grimm, V., Berger, U., Bastiansen, F., Eliassen, S., Ginot, V., Giske, J., Goss-Custard, J., Grand, T., Heinz, S.K., Huse, G., Huth, A., Jepsen, J.U., Jøorgensen, C., Mooij, W.M., Müller, B., Peer, G., Piou, C., Railsback, S.F., Robbins, A.M., Robbins, M.M., Rossmanith, E., Rüger, N., Strand, E., Souissi, S., Stillman, R.A., Vabøo, R., Visser, U., DeAngelis, D.L.: A standard protocol for describing individual-based and agent-based models. Ecological Modelling 198, 115–126 (2006)
18. Grimm, V., Railsback, S.F.: Individual-Based Modeling and Ecology. Princeton University Press (2005)
19. Grimm, V., Railsback, S.F.: Designing, formulating, and communicating agent-based models. In: Heppenstall, A.J., Crooks, A., See, L.M., Batty, M. (eds.) Agent-based Models in Geographical Systems, pp. 361–378. Springer (2012)
20. Gürcan, O., Bernon, C., Türker, K.S.: Towards a self-organized agent-based simulation model for exploration of human synamptic connections. In: CoRR 2012 (2012)
21. Haefner, J.W.: Modeling Biological Systems – Principles and Applications, 2nd edn. Springer, New York (2005)
22. Heath, B., Hill, R., Ciarallo, F.: A survey of agent-based modeling practices (january 1998 to july 2008). Journal of Artificial Societies and Social Simulation 12(4), 9 (2009)
23. Helbing, D., Balietti, S.: Agent-based modeling. In: Helbing, D. (ed.) Social Self-Organization: Understanding Complex Systems, pp. 25–70. Springer (2012)
24. Izquierdo, L.R., Polhill, J.G.: Is your model susceptible to floating-point errors? Journal of Artificial Societies and Social Simulation 9(4), 4 (2006)

25. Junges, R., Klügl, F.: How to design agent-based simulation models using agent learning. In: Rose, O., Uhrmacher, A.M. (eds.) Winter Simulation Conference, WSC 2012, Berlin, Germany, December 9-12, p. 239 (2012)
26. Klügl, F.: A validation methodology for agent-based simulations. In: Proc. of the ACM SAC Symposium "Advances in Computer Simulation, Ceara, Brasil (2008)
27. Klügl, F.: Multiagent simulation model design strategies. In: Proceedings of the Second Multi-Agent Logics, Languages, and Organisations Federated Workshops (MALLOW), Turin, Italy, September 7-10. CEUR Workshop Proceedings, vol. 494 (2009)
28. Klügl, F., Davidsson, P.: First steps towards a meta-model for mabs. In: 10th European Workshop on Multiagent Systems, Dublin (December 2012)
29. Klügl, F., Oechslein, C., Puppe, F., Dornhaus, A.: Multi-agent modelling in comparison to standard modelling. Simulation News Europe 40, 3–9 (2004)
30. Kubera, Y., Mathieu, P., Picault, S.: IODA: An interaction-oriented approach for multi-agent based simulations. Autonomous Agents and Multi-Agent Systems 23(3), 303–343 (2011)
31. Law, A.M.: Simulation Modeling & Analysis, International Edition, 4th edn. McGraw-Hill (2007)
32. Louloudi, A., Klügl, F.: Immersive face validation: A new validation technique for agent-based simulation. In: Proc. of the 6th Workshop on Multiagent Systems and Simulation (MAS&S), at FEDCIS 2012 (2012)
33. Miller, J.H., Page, S.E.: Complex Adaptive Systems – an introduction to computational models of social life. Princeton University Press (2007)
34. Norling, E., Edmonds, B., Meyer, R.: Informal approaches to developing simulation models. In: Edmonds, B., Meyer, R. (eds.) Simulating Social Complexity, Understanding Complex Systems, pp. 39–55. Springer (2013)
35. North, M.J., Macal, C.M.: Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation. Oxford University Press (2007)
36. Oechslein, C.: Vorgehensmodell mit integrierter Spezifikations- und Implementierungssprache für Multiagentensimulationen (Process Model with Integrated Specification and Implemenatation Language for Multiagent Simulations). PhD thesis, Institute of Computer Science, Universität Würzburg (2004)
37. Oechslein, C., Klügl, F., Herrler, R., Puppe, F.: Uml for behavior-oriented multi-agent simulations. In: Dunin-Keplicz, B., Nawarecki, E. (eds.) CEEMAS 2001. LNCS (LNAI), vol. 2296, pp. 217–226. Springer, Heidelberg (2002)
38. Palit, I., Phelps, S., Ng, W.L.: Can a zero-intelligence plus model explain the stylized facts of financial time series data? In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Richland, SC, vol. 2, pp. 653–660. IFAAMAS (2012)
39. Polhill, J.G.: Odd updated. Journal of Artificial Societies and Social Simulation 13(4), 9 (2010)
40. Railsback, S., Grimm, V.: Agent-Based and Individual-Based Modeling - A Practical Introduction. Princeton University Press (2012)
41. Raney, B., Voellmy, A., Çetin, N., Vrtic, M., Nagel, K.: Towards a microscopic traffic simulation of all of switzerland. In: Sloot, P.M.A., Tan, C.J.K., Dongarra, J., Hoekstra, A.G. (eds.) ICCS-ComputSci 2002, Part I. LNCS, vol. 2329, pp. 371–380. Springer, Heidelberg (2002)
42. Ricci, A., Piunti, M., Viroli, M.: Environment programming in multi-agent systems: an artifact-based perspective. Autonomous Agents and Multi-Agent Systems 23, 158–192 (2011)
43. Richiardi, M., Leombruni, R., Saam, N.J., Sonnessa, M.: A common protocol for agent-based social simulation. Journal of Artificial Societies and Social Simulation 9(1), 15 (2006)
44. Taylor, C.E., Jefferson, D.: Artificial life as a tool for biological inquiry. Artificial Life 1(1-2), 1–14 (1994)

45. Vincenti, W.G.: What Engineers Know and How They Know it. John Hopkins University Press (1990)
46. Willemain, T.: Insights on modeling from a dozen experts. Operations Research 42(2), 213–222 (1994)
47. Wray, R.E., Jones, R.M.: An introduction to Soar as an agent architecture. In: Sun, R. (ed.) Cognition and Multi-agent Interaction: From Cognitive Modeling to Social Simulation, pp. 53–78. Cambridge University Press (2005)
48. Yilmaz, L., Ören, T.: Agent-Directed Simulation and Systems Engineering. Wiley (2009)
49. Zeigler, B.P.: Theory of Modeling and Simulation. John-Wiley (1976)

Author Index