A Framework for Evaluation of Agent Oriented Methodologies*

Pedro Cuesta, Alma Gómez, Juan C. González, and Francisco J. Rodríguez

Dpto. de Informática (University of Vigo)
Ed. Politécnico, Campus As Lagoas,
Ourense E-32004 (SPAIN),
pcuesta,alma,jcmoreno,franjrm@uvigo.es
WWW home page: http://montealegre.ei.uvigo.es/gwai/

Abstract. Agent Oriented Methodologies have become an important subject of research in advanced Software Engineering. Several methodologies have been proposed, as a theoretical approach, to facilitate and support the development of complex distributed systems. An important question when facing the construction of Agent Applications is deciding which methodology to follow. Trying to answer this question, a framework with several criteria is applied in this paper for the comparative analysis of existing multiagent system methodologies. The results of the comparative over two of them, conclude that those methodologies have not reached a sufficient maturity level to be used by the software industry. The framework has also proved its utility for the evaluation of any kind of Agent Oriented Software Engineering Methodology.

1 Introduction

In the latest years, Agent Oriented Computing has emerged as an important issue in the Software Engineering field [2]. The ideas behind the concept of Agent (autonomy, social ability, reactivity/responsiveness, proactivity and intelligence: veracity, benevolence, rationality and cooperation) provide a powerful abstraction tool in the solution of complex distributed problems [12].

The interest, in the Software Engineering Approach, is focused in MultiAgent Systems (MAS) [17, 18]. A MAS is a set of autonomous agents which work cooperatively to achieve their goals. Each agent may interact with its environment or with other agents, using high level communication languages and protocols, in order to coordinate its activities and to obtain services and/or information.

Following the usual evolution in computer sciences, initially only the languages and tools which incorporate the new concepts were defined. Afterwards, the study of suitable methodologies for guiding the software development process was addressed. This evolution has implied that there are many agent oriented applications. But, until the moment, developers think that agent tools and methodologies have not achieved, the maturity level needed for being used under warranty in commercial software development.

* Research supported by the spanish national project TIC 2002-04516-C03-01
Nevertheless, agent oriented methodologies are essential in order to solve complex problems and lead the development process [10]. The construction of safe, flexible, scalable and specification adjusted MAS requires methodologies and modelling techniques which handle this complexity using the suitable supports: abstraction, structure, modularity, etc. Current Agent Oriented Software Engineering Methodologies (AOSEM) have evolved from Object Oriented Software Engineering Methodologies [7, 19] or from Knowledge Engineering Methodologies [11].

In this context we consider that the evaluation of current AOSEM will lead to important results that can afterwards be applied in the agent oriented software development: the necessity of a completely new methodology, the integration of the best characteristics of the current methodologies for obtaining a standard one, the necessity of additional documentation (samples, study cases, . . . ), etc.

This paper presents a framework for the comparative analysis of AOSEM. The proposed framework defines a set of relevant criteria for the evaluation of methodologies. These criteria have been introduced having in mind two different points of view. The first one is related to the software engineering issues (steps of development, models defined, . . . ). The other one has been the support provided to agent based orientation (definition of agent, features of communication, cooperation, . . . ). The comparison of AOSEM following these criteria will include the conceptual aspects mentioned as well as their practical application (in accordance with the documentation provided by each methodology).

Similar works [13, 15] have adapted existing frameworks in the field of classical software engineering for being used in agent methodologies. The most important drawback in the studied cases was that the defined criteria were difficult to apply in practice. These approaches introduce criteria which can only be evaluated in a subjective way. Trying to overcome these difficulties, this framework does not include subjective evaluation and introduces specific criteria for the agent oriented field. Moreover, we think that the framework could be used to evaluate future proposals; and also to compare them with the existent ones to decide whether the new methodology is useful or not in the context of agent development.

In [5] the framework has been applied to some of the most relevant proposals in AOSEM. The results obtained from this study will be used, afterwards, for achieving our global goal, that is, the definition of a methodology which integrates the best practices of these proposals and that covers the existing gap between theory and practice in Agent Applications [3, 4].

The reminder of the paper is organized as follows. First the evaluation framework is introduced and explained. After this, a case study in which we recover the evaluation results over MaSE [6, 7] and MAS-CommonKADS [9, 11] is presented. Finally, detailed conclusions about the framework and the results that can be obtained using it are shown.
2 The AOSEM Evaluation Framework

For better understanding, the criteria utilised in the evaluation framework have been classified in five groups according to their area of interest. The first group of evaluation criteria is called Development Process and incorporates general aspects of the methodology as well as other related with the stages in the construction of the system. The second one, Model View, tries to reflect the methodology concepts and their representation. Agent group, which is considered in third place, addresses the individual agent characteristics taken into account by the methodology under evaluation. Finally, the points of Additional Features and Documentation incorporate other issues of interest like the quality of the documentation provided or the extensions defined for mobility, ontologies, etc.

2.1 Development Process

In this section, aspects related to the construction of a MAS (how it is built) will be evaluated. These aspects can be summarised in the general questions: “which are the stages proposed by the methodology?” and “what kind of activities must be accomplished in each stage?”. In particular, the following aspects are considered:

- Application domain. It evaluates if the methodology is supposed to address systems from any domain or if it is domain oriented.
- Application areas. This issue collects the information presented by authors about the application of the methodology in different areas. A list of the areas is showed, underlining for each one if it is only mentioned by the authors or if the results of the methodology in this particular area are well documented (i.e. papers, manuals, books, ...).
- Open systems. It tries to evaluate if the methodology is intended or not for open systems (i.e. it allows for dynamic addition or removal of agents, or their characteristics, while MAS is running).
- Kind of lifecycle. It describes what kind of lifecycle model is applied in the development process: cascade, spiral, incremental, etc.
- Process stages. This is the main point of the evaluation; here it is described how the lifecycle is covered by the methodology. The stages of the methodology and what kind of activities must be done in each stage are presented. For addressing activities, classical software engineering taxonomy has been used: Requirements, Analysis, Design, Implementation and Testing. Requirements activities are related with the initial specification of client necessities. Analysis includes any activity related to the description of WHAT the system must do. Design activities are oriented to define HOW the system does what it must do, and how the final model of the system is obtained. Implementation activities are those related with code generation. And, finally, Testing activities try to check and validate the system under construction. Each of these activities is qualified with the term:
• “Focused”. If the methodology is fully oriented to that kind of activities in the stage under consideration.
• “Not Focused”. If the stage does not consider activities of this kind.
• “Partially Focused”, in other case.

In addition, a meaningful aspect to be considered is the implication of the user in the development process, and when this implication takes place. User implication is essential, in our opinion, in order to obtain the desired final product as it is pointed by different new software engineering methodologies [1]. This evaluation takes into account if the methodology relies on user in each stage, and the term unknown is used when this information is not clear or available.

– Supporting methodological and code generation tools. Another important issue is the availability of methodology supporting tools. The existence of tools is evaluated in relation to the activities in which the tool is used. The evaluation simply indicates if there are or not tools for each kind of activities, and how the activity is supported by the tool.

2.2 Model View

Model View section tries to evaluate the diagrams and techniques proposed by the methodology for modelling the system. The relationships between diagrams or between different views of the system, the model evolution through lifecycle and other aspects are taken into account.

– Concepts and representation. When modelling the system many concepts must be used and each of them must have a representation. This section presents a list of the concepts used and where (diagram or model) they are represented. This list is afterwards detailed for each stage of the methodology, relating the different diagrams and the concepts managed for each of them. It is though that a consistent definition of concepts through different models and a uniform representation will be a valuable feature of the considered methodology. The final point (human interaction) evaluates if human interactions with the system are specifically considered in the methodology.

– Relationships between models. The methodology may provide a way of deriving a new view of the model from an existent one, or may define rules for guaranteeing consistence and correctness when two views are related. In this point the gap between some relevant relations is underlined.

– Deliverables. In this point, the coverage provided by the different model views with respect to the whole lifecycle of the system is evaluated. By studying what are the deliverables provided by the methodology at different steps, it could be decided whether the model fulfils all the proposed stages or some of them are weakly covered. In this latest case, the models must be complemented with prototypes or with text descriptions wrote in formal, structured or natural language.
2.3 Agent

In this section, the way in which agent’s characteristics and features are defined by the methodology is taken into consideration. A fundamental point in defining MAS must be the description of the individual components of the system, that is, agents. The features considered are:

- **Concept.** The concept of agent proposed by the methodology is relevant because it introduces the conceptual framework of development. Many characteristics of individual agents may be introduced in the definition, like if the methodology is intended for a particular agent architecture (by example BDI) or if only particular kinds of agents are to be defined, etc.
- **Agent attributes.** This point is in close relation with the concept of agent and handles the intrinsic characteristics that the methodology uses: autonomy, sociability, reactivity, proactivity, intelligence or others.

In addition, the goals achieved by MAS must be more than the sum of the individual goals of the agents implied due to agents interaction and co-operation. This aspect is studied and evaluated considering:

- **Communication Types.** In this point interaction among agents is evaluated at a high level of abstraction. Methodology may allow only communication from agent to agent (A-A) (heterogeneous or homogeneous) or other kinds like human-agent (A-H) or agent-other software systems (A-O).
- **Communication Protocols.** Communication may follow well-known protocols that can be predefined in the methodology.
- **Co-operation.** This criterion shows the kinds of interaction among agents which can be defined using the methodology: negotiation, delegation of tasks, etc.
- **Agent organization.** Agents may have an internal structure with influence in the communication. This organization may be hierarchical, peer to peer, etc.

2.4 Additional Features Modelling

This point will address the extensions that normally methodologies propose to deal with important aspects of MAS, like:

- Ontological aspects
- Mobility features
- Other additional features

2.5 Documentation

An important aspect when dealing with new proposals is how they are documented. To evaluate this point the following characteristics are addressed:
Available documentation. The documentation (papers, web site, etc.) provided by authors is evaluated and qualified with: “Good”, “Sufficient” or “Poor”; where, “Poor” means that the provided documentation is not enough to make the methodology completely understandable.

Study Cases presented. It evaluates another important aspect when documenting a methodology, that is, the provided examples. It is evaluated to “Trivial” (if there is only little examples for showing particular aspects), “Partial” (in the case that examples are more complete but do not cover the whole lifecycle proposed) or “Complete” (if they cover the whole lifecycle).

3 A Case Study

Defining a framework for evaluation, as the one presented in the previous section, is useless unless it is used for comparing different methodologies. In a previous work [5] we have used the framework to evaluate the most relevant AOSEM which extend Object Oriented Methodologies, particularly Gaia, MaSE, Tropos and Message-INGENIAS. Here, the study will be centred in two different kinds of AOSEM. One of them pertaining to the class of AOSEM which are a evolution of Object Oriented Methodologies (MaSE) and the other corresponding to the field of Knowledge Engineering (MAS-CommonKADS). In this way, the capacity of the framework for being utilized for the evaluation of different kinds of Agent Methodologies is showed.

3.1 Development Process

Multiagent Systems Engineering (MaSE) [7] is a general purpose methodology for developing heterogeneous multiagent systems. MaSE covers the complete lifecycle of the system and uses a number of graphically based models, that are supported with a software engineering tool (AgentTool [6]). AgentTool also supports automatic verification of inter-agent communications and code generation. The lifecycle of MaSE is iterative. It is pretended that the analyst or designer moves among steps and phases freely such that, with each successive step, additional detail is added and, eventually, a complete and consistent system design is obtained. The purpose of the Analysis phase is to produce a set of roles whose tasks describe what the system has to do to meet its overall requirements. The goal of the Design phase is to define the overall system organization by transforming the roles and tasks defined during analysis into agent types and conversations.

MAS-CommonKADS is a multiagent methodology which extends a previous one: CommonKADS (oriented to Knowledge Based Systems development). This extension adds techniques from Object Oriented Methodologies and Responsibility Driving Design. For describing the agents protocols, methods from protocol engineering such as Specification and Description Language (SDL) and Message Sequence Charts (MSC) are used. The software process of the methodology combines the risk driven and the component based approaches in an iterative and incremental way.
Each iteration of MAS-CommonKADS is organized in the following phases: Conceptualization, Analysis, Design, Coding and Testing, Integration and Global Testing, and Deployment. During the Conceptualization phase, an elicitation task to obtain a preliminary description of the problem is carried out. The Analysis phase is oriented to get the requirements specification of the MAS through the development of different models in a risk-driven way. Analysis phase follows five steps: Agent, Task, Coordination, Knowledge and Organization modelling. To get the design model, the Design Phase consists of: Agent network design, Agent design and Platform design. The rest of the phases are expected to be approached as usually in Software Engineering Methodologies and are not explained in detail.

3.2 Model View

MaSE is based in the concept of goal. The models of analysis (Goal Hierarchy Diagram, Use Cases, Sequence Diagram, Role Model, Concurrent Task Diagram) capture the required organization, actions, and interactions among tasks, using the concepts of role and tasks. A role describes an entity that performs some function within the system. A task is a description of what the system has to do to fulfill the goals associated with a particular role. On the other hand, agent design captures roles and tasks. Roles are played by agent classes, conversations capture interaction and Actions are captured via methods. The diagrams used are Agent Class Diagram, Communication Class Diagram, Agent Architecture Diagram and Deployment Diagram. The relationship between models is well defined and the information provided is sufficient. MaSE does not explicitly model human computer interaction; it suggests that a specific role would be created to encapsulate the user interface.

The application of MAS-CommonKADS methodology is based on the development of seven different models. Each model shows the entities to be defined (called “constituents”) and the relationships among them. Each model has a textual template to define and describe each “constituent”. The models, diagrams and notation used are:

- **Agent model.** It specifies the agent characteristics using use cases, CRC (Class Responsibility Collaboration) cards, and the agent textual template.
- **Task model.** It describes the tasks that the agents can carry out: goals, decompositions, ingredients and problem-solving methods, etc. Although there is not a mandatory notation, the description of a task must include its name, a short description, input and output ingredients, task structure, ...  
- **Expertise model.** This model describes the knowledge needed by the agents to achieve their goals. It covers the development of the application knowledge (consisting of domain knowledge, inference knowledge and task knowledge) and problem solving knowledge. In order to get the desired model the Conceptual Modelling Language (CML) or the Object Model Diagrams, the inference structures and the task-method inference decomposition structures are used.
– **Organization model.** The organizational environment of the MAS and the social organization of the agent society are defined. The graphical notation of these models is based on the Object Model, adding a special symbol to distinguish agents from objects. The aggregation symbol is used for expressing agent groups.

– **Coordination model.** It models the conversations between agents: “their interactions, protocols and required capabilities”. It has two milestones: (1) definition of the communication channels and building of a prototype; (2) analysis of the interactions and determination of complex interactions (with coordination protocols). The model is presented using MSC notation, flow diagrams, state transition diagrams of SDL interactions, cooperation protocols library and HMSC (High level Message Sequence Charts) to describe and define new protocols.

– **Design model.** It consists of three submodels: “network design” for designing the relevant aspects of the agent network infrastructure; “agent design” for dividing or composing the agents of the analysis; and “platform design” for selecting the agent development platform for each agent architecture.

MAS-CommonKADS suggest to include one interface agent for each human interaction detected in the application.

### 3.3 Agent

With respect to the concept of agent, for MaSE, agents are a specialization of objects, which coordinate with each other via conversations and act proactively to accomplish individual and system-wide goals. Moreover, agents can be viewed as convenient abstractions, which may or may not possess intelligence. Designers have the choice of designing their own architecture or using predefined architectures such as BDI. Likewise, a designer may use predefined components or develop them from scratch.

MAS-CommonKADS is intended to be used to model any kind of agents; with independence of the concept of agent considered. Moreover, the methodology covers a strong concept in which the agent is an entity with mental state [16] and also a weak one, that views agents as entities that could interchange messages using a communication agent language.

### 3.4 Additional Features Modelling

Some extensions of MaSE have been proposed for integrating Ontologies [8] and for designing and specifying Mobility within the MaSE Methodology [14].

MAS-CommonKADS does not propose to include mobility aspects, but ontological aspects have been taken into account when modelling agents and their relationships.
3.5 Documentation

The principal MaSE available documentation are research papers. One important restriction of MaSE is that it does not provide a complete study case.

For MAS-CommonKADS only research papers and a Phd. Thesis are available. The travel agency case study (not fully developed) is the only application that is documented enough.

4 Conclusions

In this work, a new framework for the evaluation of AOSEM has been presented. This framework has proved its utility in the evaluation of different AOSEM, even if these methodologies are of very different nature. Particularly, the framework has been successfully applied for comparing a methodology which evolved from Object Orientation with other one that came from the Knowledge Engineering field.

In relation to the conclusions obtained from the comparison, it must be underlined that the current AOSEM are not really suitable for their application in industrial software projects, because most of them are still experimental projects. In addition, the studied proposals cover only a portion of the lifecycle or are focused exclusively on specific aspects of the development process, forgetting the rest of the activities that must be carried out.

Another important issue that must be highlighted is the lack of quality supporting tools for aiding in the system construction. Although some tools are proposed, the support provided is partial and must be further developed; particularly, the tools must be extended to handle all phases and steps, including code generation. Even more, the methodologies are presented as independent from any tool for agent development but some of them are oriented to a particular development tool.

Finally, we consider a very important restriction in the methodologies studied that they are only suitable to model close systems. Nowadays, the increasing importance of Internet applications has led to open dynamic systems. So, new proposals for been of real utility must incorporate these characteristics to systems.

References