



UNIVERSITY OF
CALGARY

**A tutorial report for SENG 609.22
Agent Based Software Engineering
Course Instructor: Dr. Behrouz H. Far**

Evaluation of Modeling Techniques for Agent- Based Systems

Prepared by: Wei Shen

ABSTRACT

To develop agent-based systems, one needs a methodology that supports the development process as common in other disciplines. In recent years, several such methodologies and modeling techniques have been suggested. An important question is, to what extent do the existing methodologies address the developers' needs. In this tutorial, I will introduce an evaluation method conducted by *Onn Shehory* and *Arnon Sturm* in their paper "*Evaluation of Modeling Techniques for Agent-Based Systems*". In the paper, the author attempt to answer this question. In evaluating existing modeling techniques, the author address criteria from software engineering as well as characteristics of agent-based systems. Their evaluation shows that some aspects of modeling techniques for agent-based systems may benefit from further enhancements. These aspects include distribution, concurrency, testing and communication richness. They also find space for (relatively small) improvements in aspects such as the refining of the models throughout the development process and the coverage and consistency checking of the suggested models.

1. INTRODUCTION

The discipline of Agent Oriented Software Engineering (AOSE) has emerged during the last decade. In this domain, methodologies and modeling techniques have been suggested in order to support the development process of agent-based systems. Both the scientific and industrial communities have recognized the potential advantages of agent-based systems. Nevertheless, the number of deployed commercial agent-based applications is not large. One of the reasons for this is the lack of mature, off the shelf, methodologies for agent based application development. As part of AOSE research we can find methodologies and modeling techniques that present concepts of software agent modeling at different levels, i.e., different lifecycle stages. The previous work mentioned above overlooked some of the software engineering aspects and agent application properties. Moreover, that work did not provide evaluation criteria for assessing advantages and drawbacks of different modeling techniques. Here, I will introduce an evaluation method conducted by *Onn Shehory* and *Arnon Sturm* in their paper "*Evaluation of Modeling Techniques for Agent-Based Systems*". In the paper, the author review some agent-oriented modeling techniques and evaluate them according to both software engineering criteria and agent-based system characteristics. From this evaluation they draw conclusions regarding the extent to which the examined agent modeling techniques address developers' needs. In addition, they examine the need for additional modeling features and extensions.

1.1 Software Engineering Evaluation Criteria

There are numerous criteria for evaluating the quality of a modeling technique from the software engineering viewpoint. Following these criteria helps in providing documentation, encouraging reuse and maintaining the development lifecycle. Among other desirable criteria, a modeling technique should adhere to the following:

1. **Preciseness:** the semantics of a modeling technique must be unambiguous in order to avoid misinterpretation of the models (of the modeling technique) by those who use it.
2. **Accessibility:** a modeling technique should be comprehensible to both experts and novices.
3. **Expressiveness** (and applicable to multiple domains): a modeling technique should be able to present:
 - The structure of the system;
 - The knowledge encapsulated within the system;
 - The data flow within the system;
 - The control flow within the system;
 - The interaction of the system with external systems.
4. **Modularity:** a modeling technique should be expressible in stages. That is, when new specification requirements are added, there is no need to modify pervious parts, and these may be used as part of the new specification.

5. **Complexity Management:** a modeling technique should be expressed, and then examined, at various levels of detail. Sometimes, high-level requirements are needed, while in other situations, more detail is required. Examination and development of all levels should be facilitated.

6. **Executability** (and testability): either a prototyping capacity or a simulation capacity should be associated with at least some aspects of the modeling technique. That is, the modeling technique has related tools that allow (possibly inefficient) computation for sample input. These should demonstrate possible behaviors of the system being modeled, and help developers determine whether the intended requirements have been expressed.

7. **Refinability** (and implementability): a modeling technique should provide a clear path for refining a model through gradual stages to reach an implementation, or at least for clearly connecting the implementation level to the design specification.

8. **Analyzability:** a methodology, or, preferably, an associated tool is available to check the internal consistency or implications of the models, or to identify aspects that seem to be unclear, such as the interrelations among seemingly unrelated operations. Such tools encourage both consistency and coverage.

9. **Openness:** a modeling technique should provide a good basis for modeling agent-based systems without coupling them to a specific architecture, infrastructure or programming language.

1.2 Agent-Based Systems Characteristics

As mentioned above, another facet of evaluating agent-based system modeling technique is the examination of agent-based system characteristics. Following are the agent-based system characteristics:

1. **Autonomy:** unlike objects, agents may be active and are responsible for their own activities: the agent has control over both its reactive and proactive behaviors. The modeling technique should support the capability of describing an agent's self-control feature.

2. **Complexity:** agent-based systems are basically sets of components (agents) that interact with each other in order to achieve their goals. These systems may consist of decision making mechanisms, learning mechanisms, reasoning mechanism and other complex algorithms. Modeling complex algorithms and mechanisms requires strong expressive power and many layers of details. A modeling technique should support such expressiveness in order to model the functionality of agent-based systems. Moreover, the complexity feature requires that modeling technique should be modular, support complexity management and describe the complex nature of an agent.

3. **Adaptability:** agent-based systems have to be flexible in order to adjust their activities to the dynamic environmental changes. The adaptability feature may require that a modeling technique be modular and that it can activate each component according to the environmental state.

4. **Concurrency:** an agent may need to perform several activities/tasks at the same time. The concurrency feature raises the requirement that some agent-based systems must be designed as parallel processing systems. This requires ability to express parallelism and concurrency in the design and implementation/deployment stages.

5. **Distribution:** multi-agent systems are sometimes working on different hosts and should be distributed over a network. This requires ability to express distribution in the design and implementation/deployment stages.

6. **Communication richness:** a basic definition of an agent consists of its autonomous activity. As such, the agent must establish communication with its environment. The environment may include other agents and information sources. The communication is characterized by its type (either interagent communication or intra-agent communication), its content and its architecture (e.g. client-server, peer-to-peer). This requires that a modeling technique should be able to express the communication characterization in order to produce agent communication command or sentences during the implementation stage.

2. A CASE STUDY AUCTION AGENT

This agent is an auction agent, that is, it participates and bids in web-based auctions on behalf of its user. To be able to use this agent for purchasing a specific item, its user must provide the agent with the following parameters: item number, user identification, private maximal price, bid step, monitoring frequency and bidding strategy and its parameters. Once activated, the agent enters the auction site and locates the specific product. Then, the agent monitors the site and retrieves the following information: leading offer (current price, user), bid step and closing date. After retrieving this information the agent parses it and acts according to the strategy that was previously selected by the user. The agent halts its auction-related activity either when its buying strategy dictate withdrawal or when the closing date of the auction passes. The agent was fully developed, that is, analysis, design, implementation and testing were all performed. The agent software components are the following:

1. The *configurator* is a GUI component that enables the agent's user to control and monitor the agent's activity.
2. The *parser* translates the information retrieved from the auction site into an internal structure.
3. The *bidder* submits bids to the auction site according to the agent's buying strategy. It implements two stages of the bidding (as required by the specific auction site): the bid and its confirmation.
4. The *manager* controls the agent's activity, monitors the auction site, activates the parser, determines the next bid as needed, activates the bidder as required and terminates the agent's purchasing activity.

3. AGENT MODELING TECHNIQUES

3.1 AOM and GAIA

Agent-Oriented Methodology (AOM) and GAIA, which extends AOM, are methodologies that concentrate on the modeling aspect of agent-based systems. AOM is based on a set of models, which are used in different stages of the agent development (both analysis and design). Following the AOM guidelines, the analysis of an intended agent-based system results in the definition of roles, which are characterized by three attributes - permissions, responsibilities and protocols - and an interaction model which depicts interactions between roles.

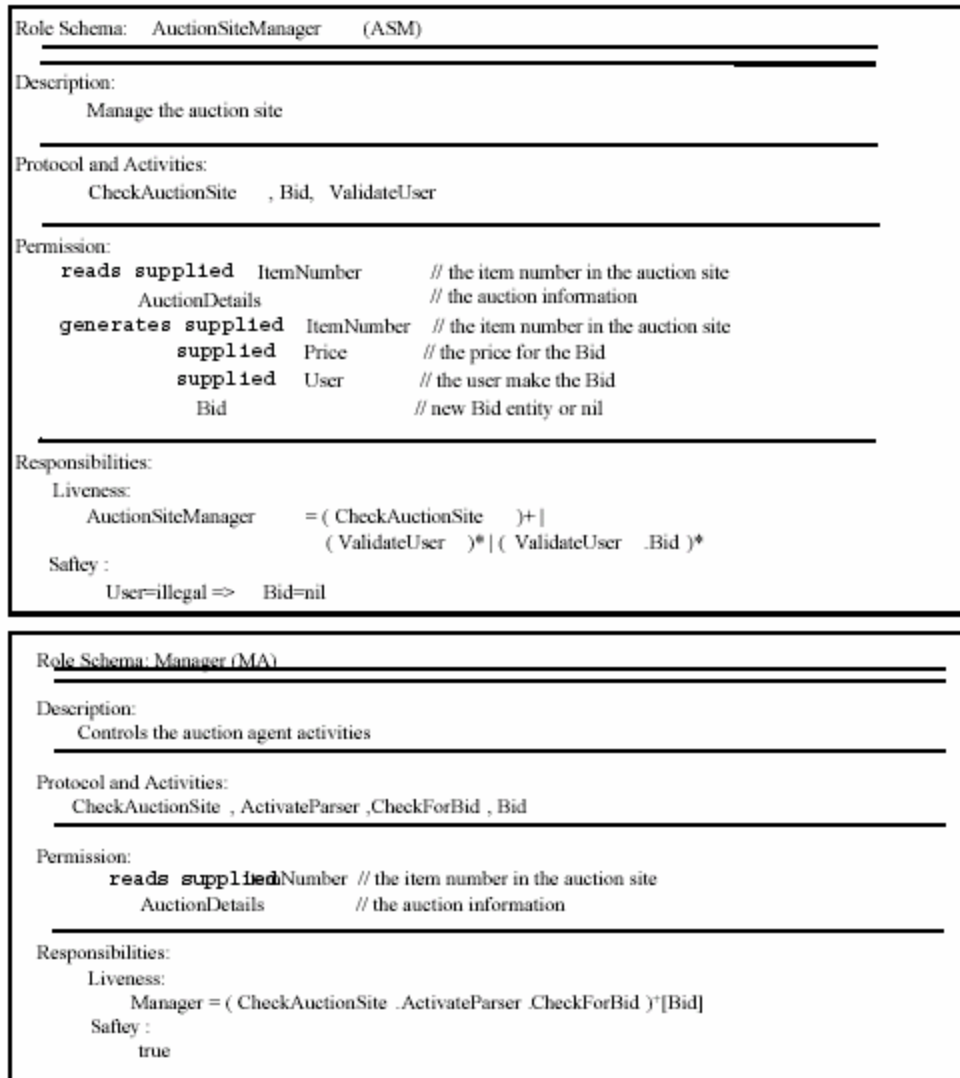


Figure 1: the Manager and the AuctionSiteManager schemata

An example of role schemata is depicted in Figure 1 and an example of the interaction model is depicted in Figure 2.

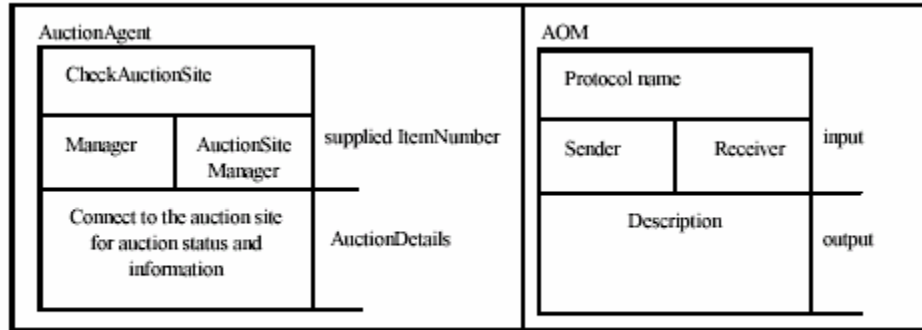


Figure 2: The Interaction Model of the CheckAuctionSite protocol

The AOM design phase consists of three models. They are agent model, service model, and acquaintance model. These models are derived from the role and the interaction models built in the analysis stage. Figure 3, figure 4, figure 5, depict these three models respectively.

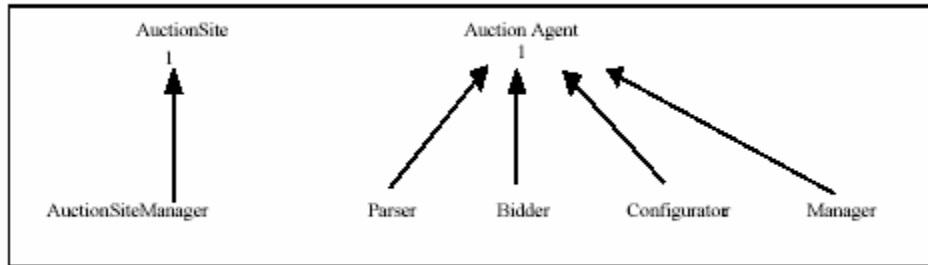


Figure 3: The Agent model represents the Auction Agent

Service	Input	Output	Pre condition	Post Condition
get auction details	ItemNumber	AuctionDetails	true	True
Validate user	User	Exist	true	(exist=true) ∨ (exist=false)
Bid	User, ItemNumber, Price	Success	user exist	(success=true) ∨(success=false)

Figure 4: The Service Model



Figure 5: The Acquaintance Model

3.1.1 Software-Engineering Evaluation

1. **Preciseness:** the liveness and safety properties, which are used for depicting the functionality of a role in a formal way (i.e., for each symbol and notation there is a clear meaning and interpretation), makes AOM accurate and prevents misinterpretation of the modeled functionality. The symbols and notations of each of the other AOM models have a clear meaning as well.
2. **Accessibility:** AOM is easy to understand and use due to its simple models and their clarity. Understanding and usage of AOM can be achieved by following the description and examples of the different models as appear in [5,6].
3. **Expressiveness:** AOM is expressive and can handle a large variety of systems due to its generic structure. However, AOM is mostly suitable for small and medium scale systems. This is because of its flatness, which limits the ability to model large amount of details. The structure of the system, the encapsulated knowledge, the data and control flows are not presented explicitly within AOM.
4. **Modularity:** AOM is mostly modular due to its design with some building blocks such as roles, protocols, activities and agent types. In AOM, one can assign new roles to agents and remove ones with no effect on the internal model of the roles. However, changes within the protocol might cause changes within the internal structure of the role. These result in changes in permissions of the role, thus limits the modularity of AOM.
5. **Complexity Management:** in AOM, there is no hierarchical presentation or any other mechanism for complexity management. The system description is flat.
6. **Executability:** this issue is not dealt with within AOM.
7. **Refinability:** this issue is not dealt with within AOM.
8. **Analyzability:** this issue is not dealt with within AOM.
9. **Openness:** AOM does not dictate any particular architecture, infrastructure or programming language. It leaves the implementation issue open for the programmer to deal with. Thus, it has an open modeling technique.

3.1.2 Agent-based System Characteristics

1. **Autonomy:** in AOM the autonomy is expressed by the fact that the role encapsulates its functionality (i.e., it is responsible for it). This functionality is internal and is not affected by the environment, thus represents the role's autonomy. In addition, in AOM one can model alternative computational paths, which gives the role (and agents that consist of this role) the autonomy in making decisions.
2. **Complexity:** in AOM one can describe the agent's complexity by integrating several models: the role model, the acquaintance model and the agent model. However, it is difficult to model some procedural processes and complex computation in AOM.
3. **Adaptability:** adaptability in AOM can be expressed by the liveness properties. The liveness property operators within AOM symbolize optional execution, thus expresses the potential adaptability of the agent to environmental changes (although the changes are not described explicitly).
4. **Concurrency:** this issue is not dealt with within AOM.
5. **Distribution:** AOM allows the expression of distribution via the acquaintance model, where each role can be performed on any host or processor. Yet, it does not explicitly address the distribution issue.

6. Communication richness: the communication aspects that are dealt with within AOM are the protocols and the communication paths between agents. However, the content of a message and the communication architecture are not expressed in any of the AOM models.

3.2 ADEPT

The Advanced Decision Environment for Process Tasks (ADEPT) system is a complete infrastructure for designing and implementing multi-agent systems. As such, ADEPT supplies a set of tools (i.e., models and a language) to achieve its purpose. In ADEPT, the main building block (from the modeling viewpoint) is the agency. An agency is recursively defined and consists of a single responsible (or controlling) agent, a set of tasks and a set of sub-agencies. The tasks that the agent is responsible for can be viewed as simple services with a well-defined input, output, guard and functional specifications. An ADEPT-specific language called SDL (Service Description Language) defines the services and the information of the agent. In addition, ADEPT has a negotiation model, which includes a protocol, a service level agreement (SLA) and a reasoning model. The protocols are relatively standard and are based on speech-act theory. To communicate, agents in ADEPT need a protocol and a SLA. The latter is a template that defines the agreement type. One of the fields of the SLA indicates the service associated with it. Both the client and the server agents should know this template. The reasoning model consists of two knowledge-based components: a declarative one and a procedural one. The declarative one sets the negotiation context and the procedural one specifies which action should be taken given the declarative knowledge. The procedural knowledge base is represented as a set of strategies and mechanisms for selecting between them.

Figure 6 depicts an ADEPT model of the case study Auction Agent. The Figure presents two agents associated with their agencies and tasks. The Auction Agent is responsible for the agency, which should perform the tasks of managing, bidding, configuring and parsing. A rounded rectangle indicates a responsible agent, a box indicates an agency, diamond denotes a task and the arrows indicate the services supplied by the agent. The specification of the information encapsulated within the agents is depicted in Figure 7 and the service specification of the *Get_Auction_Details* service is depicted in Figure 8.

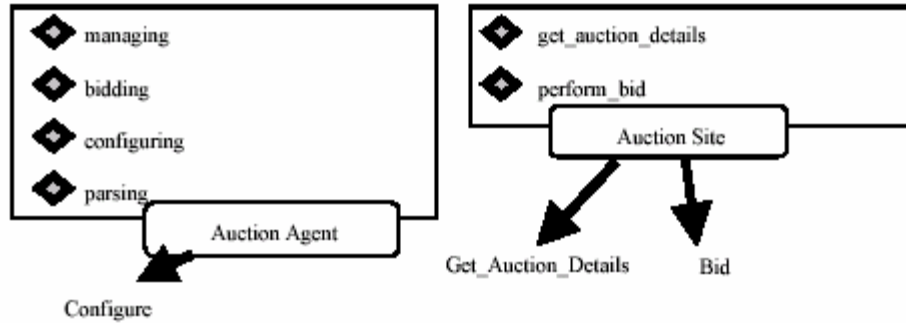


Figure 6: The Agent-Agencies Model

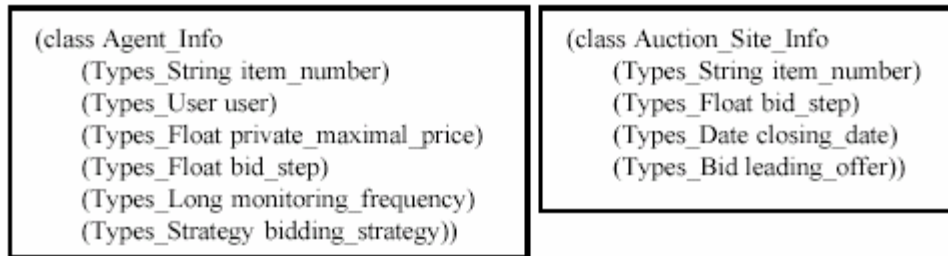


Figure 7: SDL description of an information object

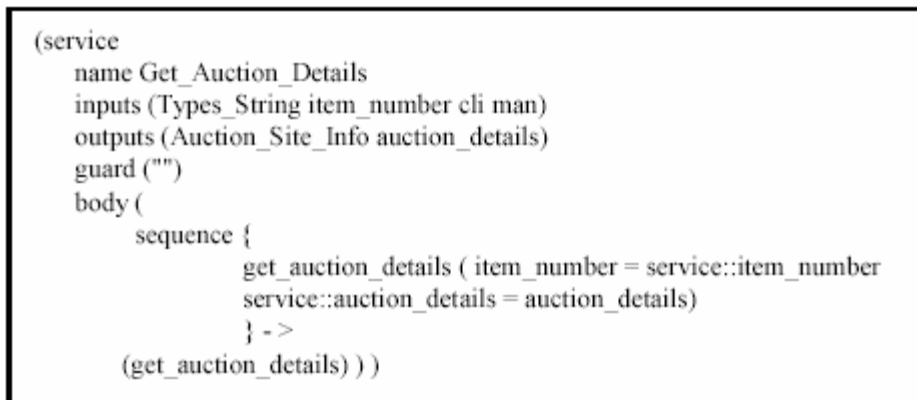


Figure 8: SDL description of the Get_Auction_Details service provided by the Auction Site agent

3.2.1 Software-Engineering Evaluation

1. **Preciseness:** a formal specification of SDL is provided. Other formal specifications of ADEPT have not been published yet however exist. These formal specifications prevent misinterpretation of ADEPT models.
2. **Accessibility:** learning, understanding and implementing ADEPT models is not difficult. The majority of the ADEPT models are intelligible and provide a clear notations and semantics. However, the reasoning model might introduce some difficulties in its understanding and assimilation, due to its complex logic.
3. **Expressiveness:** integration of the models supported by ADEPT provides an expressive power, especially for business processes. These models represent the structure of the system (sometimes a reflection of the organization structure), the behavior of the

system and its response to environmental changes. The data and control flow can be analyzed and understood from the SDL and the reasoning model, but they are not presented explicitly. This implies difficulty for the designers and implementers in modeling and understanding the data and control flow. In addition, SDL represents the knowledge structure within the system. An implementation of an ADEPT agent or MAS is coupled to the ADEPT architecture, which means that it uses the ADEPT modules (Such as Service Execution Module and Communication Module). These modules execute the different models mentioned before. Hence, there is no need for defining the system structure.

4. **Modularity:** the use of agencies and agents increases the modularity, as agencies can be easily moved from one agency to another.

5. **Complexity Management:** hierarchical presentation and complexity management can be achieved by using the agent and agencies model, which enables (de-) composition of agents and agencies.

6. **Executability:** ADEPT has its own engine, which means that once the modeling phase was finished the agent can be executed. However, it is not clear how much effort the designer has to invest in the modeling phase (analysis and design) in order to get the agent to run.

7. **Refinability:** ADEPT has its own definition language (SDL), which defines the services derived from the model of agents and agencies. This means that the model of agents and agencies can be refined by SDL.

8. **Analyzability** the consistency and coverage checks are not dealt with within ADEPT.

9. **Openness:** ADEPT is a closed system. The different models are tightly coupled to the ADEPT architecture.

3.2.2 Agent-based System Characteristics

1. **Autonomy:** the autonomous nature of an agent is modeled by encapsulating tasks within an agency. This means that agent that controls the agency is autonomous in the timing and execution of the encapsulated tasks. The decision-making aspect of the agent's autonomy is modeled by combining the service model and the reasoning model within ADEPT. These models describe the agent's activities upon environment and the agent's state changes.

2. **Complexity:** by using the ADEPT models one can present large and complex systems. However, the tools supplied by ADEPT are not powerful enough to express complex algorithms.

3. **Adaptability:** the reasoning model expresses the adaptability in ADEPT. The reasoning model has the capability of representing several states regarding the negotiation context and additional rules. The negotiation context and additional rules provide the agent with guidance regarding the activities it should perform. This enables the agent to adapt to environmental and state changes.

4. **Concurrency:** is not dealt with within ADEPT from the modeling viewpoint.

5. **Distribution:** is not dealt with within the ADEPT from the modeling viewpoint.

6. **Communication richness:** the communication between agents is handled by the system architecture, but its modeling is unclear.

3.3 DESIRE

The Design and Specification of Interacting Reasoning (DESIRE) framework is a complete environment for design and implementation of MAS. It allows the system designer to explicitly and precisely specify both the intra-agent and inter-agent functionality. In DESIRE, the following models are supported: (1) task (de-) composition, (2) information exchange, (3) sequencing of (sub-) tasks, (4) subtask delegation and (5) knowledge structure.

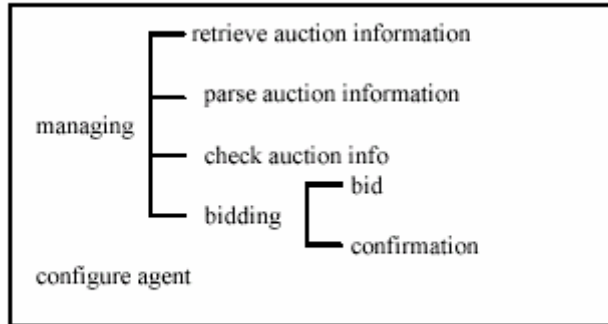


Figure 9: Task Hierarchy of the Auction Agent

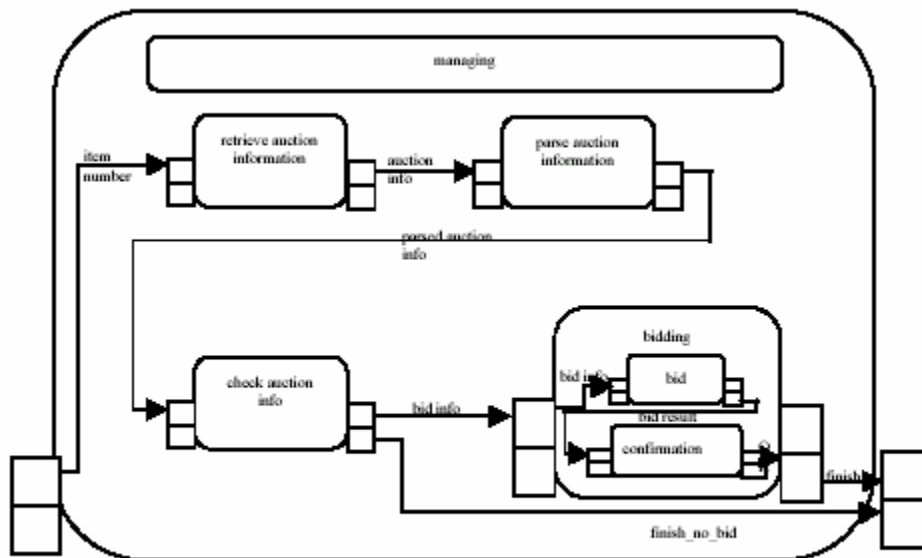


Figure 10: The component managing of the Auction Agent

In DESIRE, the information exchange between tasks is specified as information links between components. Each information link directs the output of one component to the input of another one. The information exchange model is depicted in Figure 10. This Figure shows the task hierarchy that was previously presented in Figure 9. The additional information is the information links that depict the data flow. For example, the link *ItemNumber* indicates that there is an input to the *retrieve auction information* task. Its output is fed as input, via the *auction info* link to the *check auction info* task. Note that the information exchange model has a formal syntax. Task sequencing is explicitly modeled within the component as task control knowledge, which can be seen in Figure

11. This knowledge includes the order of subtasks, their activation target (usually referred to as a goal in the context of agents) and the amount of effort, which can be afforded. The target is the focus of the activity of the component and the effort is the component and the link states. For example, if the task *parse_auction_information* succeeds then the next task should be *check_auction_info* and its target should be *get_bid_info*. In addition, DESIRE awakes the relevant links that should be activated if the *check_auction_info* task is successful. DESIRE has the additional advantage that the specifications and their semantics can be expressed formally, using temporal logic.

```

1 fcomponent-state(managing,start)
then next-component-state(retrieve_auction_info,active)
  and next-target-set(retrieve_auction_info,get-info)
  and next-link-state(item_number,awake)
  and next-link-state(auction_info,awake)

1 fevaluation(retrieve_auction_info,get-info,succeed)
then next-component-state(parse_auction_info)
  and next-target-set(parse_auction_info, get_info)
  and next-link-state(parsed_auction_info,awake)

1 fevaluation(parse_auction_info,get-info,succeed)
then next-component-state(check_auction_info)
  and next-target-set(check_auction_info, get_bid_info)
  and next-link-state(bid_info,awake)
  and next-link-state(finish_no_bid ,awake)

1 fevaluation(check_auction_info,get_bid_info,succeed)
then next-component-state(bid)
  and next-target-set(bid, place_bid)
  and next-link-state(finish,awake)

```

Figure 11: The task control knowledge of managing

3.3.1 Software-Engineering Evaluation

1. **Preciseness:** the DESIRE infrastructure is based on temporal logic, which is precise and prevents misinterpretation of the modeling outcomes.
2. **Accessibility:** DESIRE has a wide range of modeling capabilities. However, this richness results in difficulty to learn and implement it.
3. **Expressiveness:** DESIRE can express many application domains using and integrating its models. The data flow modeling within DESIRE is performed using the links in the task composition model. The control flow can be understood from the task control knowledge. The data modeling is done using the knowledge structure mechanism. An implementation of a DESIRE agent or MAS is coupled to the DESIRE architecture, which means that it uses the DESIRE infrastructure. This infrastructure supports the execution of the modeled system. Hence, there is no need for defining the system structure.

4. **Modularity:** DESIRE supports modularity within the component model. This means that components can be changed and replaced without any effect on others.
5. **Complexity Management:** the hierarchical presentation and complexity management is achieved within the models of task hierarchy and components.
6. **Executability:** DESIRE has its own engine. After the modeling is finished, the modeled system should run. DESIRE has an automated implementation generator, however, the designer cannot determine the implemented software architecture.
7. **Refinability:** the DESIRE models are not formally assigned to a specific development stage (i.e., analysis, design, and implementation). However, DESIRE enables to refine the models at any stage.
8. **Analyzability:** correctness and coverage are checked using the formal semantic specification.
9. **Openness:** DESIRE is an open system in the sense that it is not coupled to a specific programming language or architecture.

3.3.2 Agent-based System Characteristics

1. **Autonomy:** the autonomy of an agent can be expressed within the DESIRE task control knowledge. In this model, one can track the possible execution paths of the agent activities regardless of the environment. That is, the agent has its autonomy of executing the tasks. Also, one can model a decision mechanism of the agent, which reflects the autonomy of the agent from the decision-making aspect.
2. **Complexity:** DESIRE models enable one to describe the required complexity of a software agent. However, (according to the reviewed literature) DESIRE does not provide tools for modeling complex computation.
3. **Adaptability:** the flexibility of the agent is modeled within the task control knowledge. In this model the agent's behavior can change according to new goals and other environmental changes.
4. **Concurrency:** the concurrency issue is dealt with from the functional point of view but not from the execution point of view. This means that one can model concurrency within the task control knowledge but the implementation of this concurrency is not explicitly addressed.
5. **Distribution:** the distribution issue in DESIRE can be expressed using the task composition model, where each task can be performed on any host or processor. However the distribution issue is not addressed explicitly in DESIRE.
6. **Communication richness:** the communication in DESIRE is partially dealt with by presenting the links between components. However, the DESIRE literature does not refer to a message content and means.

4. CONCLUSION

Table 1 provides a qualitative summary of the evaluation. Each modeling technique is evaluated by indicating to what level it addresses each criterion. In general, the modeling techniques adequately addressed agent-based system characteristics. Yet, the author's conclusions regarding software engineering issues suggest that AOSE still has a way to go to provide industrially applicable modeling technique.

Although agent-based system characteristics are usually properly addressed, a few issues can benefit from being dealt with: Distribution and Communication richness.

To conclude, the current agent-oriented modeling techniques already provide a wide array of features advantageous for agent modeling. However, there is a need for further exploration of the issues mentioned above. Based on these findings, the author intends in future research, to address the needs of agent-based system developers. This should be done in order to find the required modeling techniques and components for building agent-based systems.

Table 1. Modeling Techniques Evaluation Summary

Methods Criteria	AOM	ADEPT	DESIRE
Preciseness	*	*	+
Accessibility	+	-	*
Expressiveness	*	+	+
Modularity	-	+	+
Complexity Management	-	+	+
Executability	NS	+	*
Refinability	NS	*	*
Analyzability	NS	NS	+
Openness	+	-	+
Autonomy	*	+	+
Complexity	-	*	*
Adaptability	*	*	+
Concurrency	NS	NS	*
Distribution	NS	NS	NS
Communication richness	-	-	-

+ good, * satisfying, - dissatisfying, NS - not supported

5. My Comments

In our group project, our group used GAIA to identify the roles and design the protocols. I found that GAIA has the following shortcomings:

- 1) The roles are identified by the functional requirements. The nonfunctional requirements such as performance, security, fault tolerance, accuracy, maintainability that play a prominent role in architectural design are overlooked.
- 2) When we were doing the detailed design of each agent. I found there are big overlaps between the GAIA (Roles schema, Protocol model, the

service model) and the Rational Rose diagram (Use case chart, Use case definition, sequence chart). We just do some repeat works. Since Rational Rose is so popular in the designation of software. I question the usability of GAIA in the real software development. I think the Agent-based methodologies can be used at the analysis phase to identify the roles and agents. Once the analysis has been accomplished, the design and implementation can be based on other methodologies such as object-oriented design.

- 3) GAIA does not provide any method about how to design the system architecture. During the designation of our project, I found the designations of the system architecture and data structure are critical to meet the system's nonfunctional requirements. Especially for a real-time system like our project topic, proper designation of the system architecture and the data structure will improve the system's performance dramatically.
- 4) GAIA does not provide any principles about how to identify roles and agents. Should we minimize the number of agents? Can we just identify the agents by their physical place in the proposed system and put all the roles into one agent in a place in order to reduce the communication overhead between agents?
- 5) How to design and organize the knowledge base is missing in GAIA.

The rest two methodologies also have the same problems except that they provide the system architecture. I think there are pros and cons for a methodology coupled to a specific system architecture. On one side, there is no need for defining the system structure, thus saving some effort for the system designers. On the other side, coupling to a specific architecture make this methodology lack of flexibility. Different type of system may require different type of architecture. From this perspective, GAIA has its advantage.

I found a contradiction in this paper. When evaluating the DESIRE by using the **Expressiveness** criteria from the software-engineering viewpoint, the author think an implementation of a DESIRE agent is coupled to the DESIRE architecture. Nevertheless, when evaluating the DESIRE by using the **Openness** criteria, the author think DESIRE is an open system in the sense that it is not coupled to a specific programming language or architecture. These two conclusions seem to be contradicting with each other.

6. REFERENCES

- [1] Onn Shehory, Arnon Sturm, Evaluation of Modeling Techniques for Agent-Based Systems, Proceedings of the fifth international conference on Autonomous agents May 2001
- [2] M. A. Ardis, J. A. Chaves, L. J. Jagadeesan, P. Mataga, C. Puchol, M. G. Staskauskas and J. Von Olnhausen, A Framework for Evaluating Specification Methods for Reactive Systems, Experience Report, Proc. of 17th Intl. Conf. on Software Engineering, 1995.
- [3] B. Bauer, J.P. Muller, and J. Odell, Agent UML: A Formalism for Specifying Multiagent Software Systems: P. Ciancarinin and M.J. Wooldridge (eds.) Agent-Oriented Software Engineering, LNCS 1957, Springer, 2000, pp. 91-103.
- [4] F. M. T. Brazier, B. Dunin-Keplicz, N. R. Jennings and J. Treur, DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework. Intl. Journal of Cooperative Information Systems, vol. 6, Formal Methods in Cooperative Information Systems: Multi-Agent Systems, (M. Huhns and M. Singh, eds.), 1997, pp. 67-94.
- [5] M. Wooldridge, N. R. Jennings, and D. Kinny, A Methodology for Agent-Oriented Analysis and Design Proc. of Agents-99, Seattle, WA, 1999, 69-76.
- [6] M. Wooldridge, N. R. Jennings, and D. Kinny, The Gaia Methodology for Agent-Oriented Analysis and Design, Journal of Autonomous Agents and MAS 3 (3), 2000, 285-312.