

# An Investigation of Agent Oriented Software Engineering Methodologies to Provide an Extended Methodology

A.Fatemi<sup>1</sup>, N.NematBakhsh<sup>2</sup>, B. Tork Ladani<sup>3</sup>

Department of Computer Science, Isfahan University, Isfahan, Iran

<sup>1</sup>Afsaneh\_Fatemi@hotmail.com, <sup>2</sup>Nemat@eng.ui.ac.ir, <sup>3</sup>Ladani@eng.ui.ac.ir

## Abstract

*The area of agent-oriented methodologies is maturing rapidly and the time has come to begin drawing together the work of various research groups with the aim of developing the next generation of agent-oriented software engineering methodologies. An important step is to understand the differences between the various key methodologies, and to understand each methodology's strengths, weaknesses, and domains of applicability. In this paper we perform an investigation upon user views, on four well-known methodologies. We extend Tropos, as the most complete one up on users view point, by providing a proper supportive tool for it.*

## 1. Introduction

One of the most fundamental obstacles to large-scale take-up of agent technology is the lack of mature software development methodologies for agent-based systems [1].

Even though many Agent Oriented Software Engineering (AOSE) methodologies have been proposed, few are mature or described in sufficient detail to be of real use. In fact, the area of agent-oriented methodologies is maturing rapidly and that the time has come to begin drawing together the work of various research groups with the aim of developing the next generation of agent-oriented software engineering methodologies [1,5].

An important step is to understand the differences between the various key methodologies, and to understand each methodology's strengths, weaknesses, and domains of applicability.

In this paper we perform a comparison upon users reviews, on four well-known methodologies: Gaia, Message, Prometheus and Tropos .

Five sample systems were analyzed, designed and implemented, each using all of these methodologies, by a number of graduate students. The purpose of this experiment was not to provide statistically significant

results. The main objective of this part of research was to examine agent-oriented methodology's ability to solve a small real problem. So, we found whether the methodology is understandable and usable.

In a higher stage, we used the results of comparison to choose the best methodology based on the users view points, and extend it to provide an improved methodology.

In section 2, we briefly introduce the four investigated methodologies. It is impossible to have more details, so we refer the reader to original sources for further details on each methodology. In section 3 we present our findings about the strengths and weaknesses of each methodology from the view point of its users. Here we would be able to compare the methodologies and select the best from the users view point in section 4. In Section 5 we propose our extension to selected methodology. We conclude the paper and offer directions for future work in section 6.

## 2. The Methodologies

### 2.1. Gaia

Gaia is one of the first methodologies which is specifically tailored to the analysis and design of agent-based systems. Its main purpose is to provide the designers with a modeling framework and several associated techniques to design agent-oriented systems. Gaia separates the process of designing software into two different stages: analysis and design. Analysis involves building the conceptual models of the target system, whereas the design stage transforms those abstract constructs to concrete entities which have direct mapping to implementation code [9].

The main artifacts of Gaia are Role Model and Interaction Model (Analysis), and Agent Model, Services Model, and Acquaintance Model (Design) [10].

### 2.2. Message

Message is the end product of a two-year project hosted by the European Institute for Research and Strategic Studies in Telecommunications (EURESCOM). The main purpose of the project is to develop an agent-oriented development methodology as an extension to existing methodologies to allow them to support agent-oriented software engineering [7].

The current status of the methodology is, however, limited to analysis and design activities. Additionally, the Unified Modeling Language (UML) is selected as a foundation for Message's modeling language and is extended by adding entity and relationship concepts required for agent-oriented modeling.

Message is developed based on a view-oriented approach. The full analysis model is described via five different views: Organization view (OV), Goal/Task view (GTV), Agent/Role view (AV), Interaction view (IV) and Domain view (DV). In contrast to the analysis phase where process steps and their artifacts are clearly defined, the Message design stage is not well-documented [7].

The authors of Message argued that this provides flexibility in the sense that the designers are allowed to choose among different design approaches.

### **2.3. Prometheus**

The Prometheus methodology is a detailed AOSE methodology that is aimed at non-experts. It has been successfully taught to and used by undergraduate students. Prometheus consists of three phases: System specification, architectural design, and detailed design [11].

The first phase of Prometheus, the system specification phase, involves two activities: determining the system's environment, and determining the goals and functionality of the system.

The second stage, architectural design, involves three activities: defining agent types, designing the overall system structure, and designing the interactions between agents.

The internals of each agent and how it will accomplish its tasks within the overall system are addressed in the detailed design phase. It focuses on defining capabilities, internal events, plans and detailed data structure for each agent type identified in the previous step [11,12].

### **2.4. Tropos**

Tropos is an agent-oriented software development methodology created by a group of authors from various universities in Canada and Italy [2]. One of the significant differences between Tropos and the other methodologies is its strong focus on early requirements analysis where the domain stake-holders and their intentions are identified and analyzed. This analysis process allows the reason for developing the software to be captured. The software development process of Tropos consists of five phases: Early Requirements, Late Requirements, Architectural Design, Detailed Design and Implementation [4,5,8].

## **3. Features of methodologies according to user**

In this section, we briefly describe a methodology evaluation upon user reviews within which the methodology comparison is conducted.

Five sample systems were analyzed, designed and implemented, each using all of these methodologies, by a number of graduate students. The purpose of this experiment was not to provide statistically significant results. The main objective of this research was to examine agent-oriented methodology's ability to solve a small real problem. So, we found whether the methodology is understandable and usable. The experimental application that we used in this paper as a case study, was an On-line Student Registration system. Its main goal is to assist a student in on-line secure reliable registering.

The Methodologies evaluation results from users' view point are described below:

### **3.1. Gaia**

Gaia analysis phase includes constructing a role model and protocol diagrams. In order to do so, Gaia suggests the analyst follow the three process steps: make the prototypical roles model, make the protocol diagrams and make the elaborated roles model.

According to the user, the stages are generally well-described in the related journal paper. However, the methodology seems to lack support for helping the analysts identify roles in the system.

In addition, there are several notations which need to be explicitly defined. The analysis process requires many iterations, which resulted in issues such as model consistency, links between models and keeping track of changes made. Those problems were manifested by the lack of tool support (no tool support for Gaia).

Additionally, it seems that Gaia is quite general and lacked detailed design supports. As a result, it is difficult to move from the design stage to the implementation phase based upon the artifacts she currently created.

### 3.2. Message

According to user, the analysis stage is well defined and documented. In addition, the view-oriented approach which is used in Message is very suitable because it is easy to analyze the system by focusing on different aspects of the system at different times. The five different views are equally important and complete, i.e. together they provide the comprehensive view of the system and its environment.

One of the important issues in the view-oriented approach is to deal with the maintenance of consistency among views. Relating to this, tools supporting the methodology play an important role. The tool support recommended by Message (MetaEdit) is only limited to drawing diagrams and generating report documentation and it is a negative point.

The refinement-based approach which the Message analysis process applies to is very useful. It guides designer from the beginning where he just had a little idea of the system to the end where components of the system are described in detail.

There is not enough documentation about the design stage. Its purpose is to let the designers have the freedom to choose the design that suits their specific agent platform, but it makes this stage hard to perform [7].

Overall, the user has a good impression of the methodology. It can be because of closeness of Message model concepts to UML. Also, to some extent, Message helps designer think, analyze and design the system in an agent-oriented manner. The analysis stage is the strength of the methodology whereas the capturing requirements phase, design and implementation phases need to be described at sufficient level of detail.

Message does not provide a detailed design nor implementation, which makes implementation a very difficult task.

### 3.3. Prometheus

Overall, users have a good impression on Prometheus. Despite the fact that goals are a new concept introduced in the agent-oriented paradigm, it is easy for user to identify system goals. However, determining the external interface (percepts, actions,

data stored) is harder for user. In particular, there are some difficulties in identifying the relationship between functionalities and actions as well as telling the difference between actions and sending messages.

There are some difficulties in understanding and using several concepts introduced at the system specification phase. These difficulties may be caused by inconsistent descriptions of these concepts in different documentation related to Prometheus.

In addition, the Prometheus Design Tool (PDT) is introduced as a support tool for it that can be used in drawing design diagrams, checking consistency among design models and automatically generating reports. Also regarding tool support, there is several difficulties and complications in developing interaction diagrams and protocols. They are in fact caused by the lack of support for drawing these diagrams in PDT.

### 3.4. Tropos

The designer using Tropos, appreciates the usefulness of its requirements analysis phase (including Early Requirements and Late Requirements). The goal-driven requirements engineering techniques described in this phase helps him in establishing an agent-oriented way of analyzing and designing the target system. In fact, the concepts described in this phase, such as actors, goals, plans, etc, are very similar to those in agents. This early requirement analysis phase also involves identifying goals of the target system and the resources, plans and tasks to achieve these goals. It requires the user to reason about the system at a higher level of abstraction compared with the object-oriented methodologies.

Tropos architectural phase is similar to its counterpart in the object oriented methodology. Tropos divides this stage into three process steps: defining the overall architectural organization of the system, identifying the capabilities, and defining agent types and mapping them to capabilities. The last step is the most important in this stage from the user's view point. Even though, there is not enough detailed information available describing it.

There are some difficulties in grouping capabilities into agent types.

Several Supportive tools are introduced for Tropos, but none of them can support all five phases of it. Jack is one of these tools that can support phases 3 to 5 [3]. We proposed a heuristic for mapping all Tropos concepts to Jack design elements. In this manner, Jack can be used as a complete tool to support all Tropos phases, from early requirement analysis to implementation and code generation.

## 4. Comparing methodologies

Based on the above described features, we have performed a brief comparison between selected methodologies that is explained below.

With regard to agent-oriented concepts, the level of support for autonomy of all of the methodologies is overall good. Prometheus and Tropos support very well the use of mental concepts (such as beliefs, desires, intentions) in modeling agents' internals whereas Message and Gaia provide weaker support. In addition, Tropos supports pro-activeness and re-activeness more than the others.

According to users, the concepts used in the methodologies tend to be clearly explained and understandable. However, Prometheus has some confusing terminology such as the distinction between percept, incident, trigger, and event.

All four methodologies were perceived as being clearly agent-oriented.

Overall, the users felt that the methodologies' notations were clear and well defined (syntax/semantics) and easy to use. Tropos is an interesting case: there is disagreement on whether the concepts be clear, and whether the notation be clear and easy to use.

In terms of consistency checking, Prometheus supports it well whereas the others do not appear to support it. Refinement, modularity, and hierarchical modeling are generally supported (Gaia has the least and Prometheus has the best support), however reuse is not well handled by any of the methodologies.

Overall, the users had a very good impression of the notation of all the methodologies. They also reported some minor issues that are notified in chapter 3 sub sections.

From the software development life-cycle point of view, all of the methodologies cover the requirements, architectural design and detailed design. The analysis stage of all methodologies is well described and provided useful examples with heuristics. This helps users to shift from object-oriented thinking to agent-oriented. The implementation phase is not well supported; Only Tropos briefly explains that the concepts developed during design map to Jack constructs but does not provide a detailed process, heuristics, examples, or a discussion of the issues. Only Prometheus mentions testing/debugging in a low level.

From the practical usability view point, Prometheus targets undergraduate and industry programmers,

whereas Tropos is aimed at experts. Gaia and Message are not used practically in many cases.

Furthermore, from the complexity of the methodology to users point, it seems that Message is the easiest because of similarity of its design language to UML. Regarding the availability of resources supporting the methodologies, Prometheus is the best.

None of the methodologies seem to address issues such as quality assurance, or cost estimating guidelines. The availability of tool support also varies. Prometheus is well supported with JDE and PDT. Tropos can be supported with Jack partially. As mentioned earlier, we proposed a mapping method to use Jack as a complete tool for Tropos.

MetaEdit is a weak support for Message, and Gaia suffers from the lack of supporting tool.

According to above features, we select Tropos as the best methodology upon our user view points, and try to extend it towards a complete useful development methodology.

## 5. EJack<sup>1</sup>: A complete supportive tool for Tropos

As mentioned earlier, Tropos is one of the most accurate and complete agent oriented methodologies that covers software development from early requirement analysis to implementation. Several supportive tools are introduced for Tropos, but lacking a proper tool to cover all five phases is one of the most essential deficiencies of Tropos. Table 1 shows some of existing tools and phases they support [2].

**Table 1. Tropos tools and their supported phases**

Tool	Supported Phases
TAOM4E	1,2
UML	1,2,3,4
Jack	3,4,5

It can be concluded from Table 1 that UML can support Tropos through changing stereotypes, but in this way, some of the diagrams of phases 1 and 2 can not be presented. In addition, it can be concluded that TAOM4E and Jack together, can support all phases of Tropos.

In this part, we propose a method to use existing Jack development tool, as a complete tool for Tropos. We do this through mapping all concepts of Tropos to

---

<sup>1</sup> Extended Jack

existing Jack concepts resulting EJack development tool. EJack seems to be an appropriate tool to support Tropos, due to experimental results.

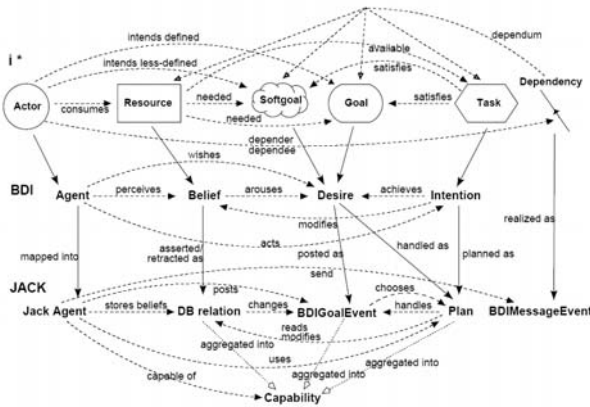
### 5.1. Jack and EJack

Jack is a BDI<sup>2</sup> agent-oriented development environment built on top and fully integrated with Java, where agents are autonomous software components that have explicit goals (desires) to achieve or events to handle. Agents are programmed with a set of plans in order to make them capable of achieving goals [3].

To support the programming of BDI agents, Jack offers five principal language constructs: Agent, Capability, Belief, Event, and Plan [6].

On the other hand, Tropos adopts the *i\** modeling framework [13], which offers the notions of actor, goal and (actor) dependency, and uses these as a foundation to model early and late requirements, architectural and detailed design.

Figure 1 summarizes the mapping from *i\** concepts to Jack constructs and how each concept is related to the others within the same model [6].








**Figure 1. Converting Tropos concepts to Jack's, using BDI model concepts**

Since representation of agent beliefs is one of the Tropos objectives, relation between Jack and Tropos can be understood clearly.

The diagrams produced in stages 1 and 2 by TAMO4E, should be converted to include Jack concepts, because of the difference between Jack and Tropos in key concepts. So, according to figure 1, we first convert Tropos concepts to BDI concepts and then

change them to Jack concepts. This mapping, that is the basis of EJack, is shown in table 2.

**Table 2. Mapping between Tropos and Jack concepts to construct EJack**

Tropos	Jack
Actor	 Agent
Resource	 Named Data
Task	 Plan
Soft Goal & Hard Goal	 Event
Actor Capability	 Capability

### 5.2. A Case Study

The experimental application that we used in this paper as a case study, is an Student e-Registration system. Its main goal is to assist a student in on-line secure reliable registering.

The student logs in to system and Select his/her required courses/sections, after visiting courses/sections lists. Course preferences, section capacities, and so on should be meted to acceptance of a course/section. After selecting all required courses, the student should confirm his/her registration. Confirmation, also, needs to meet some preferences.

Figure 2 shows the architectural design diagram, one of the designed diagrams, due to Tropos concepts. This diagram can be generated in EJack due to Jack development concepts. Figure 3 shows the architectural design diagram generating with EJack

<sup>2</sup> Belief, Desire, Intention

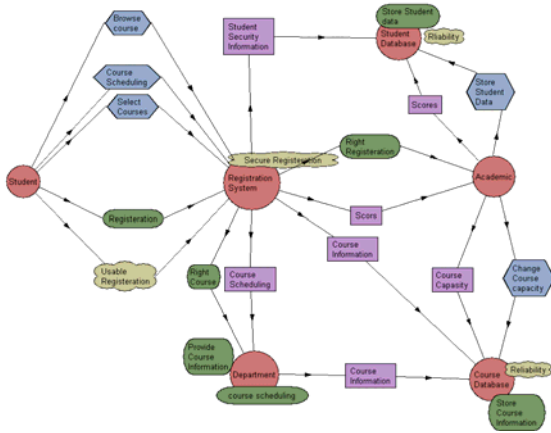


Figure 2. The architectural design diagram



Figure 3. The architectural design diagram generated by EJack

## 6. Conclusion and future works

One of the most fundamental obstacles to large-scale take-up of agent technology is the lack of mature software development methodologies for agent-based systems.

Even though many Agent Oriented Software Engineering (AOSE) methodologies have been proposed, few are mature or described in sufficient detail to be of real use.

An important step towards a complete unique methodology, is to understand the differences between the various key methodologies, and to understand each methodology's strengths, weaknesses, and domains of applicability.

In this paper we performed a comparison upon users reviews, on four well-known methodologies: Gaia, Message, Prometheus and Tropos .

We selected Tropos as one of the most accurate and complete agent oriented methodologies upon our comparison and tried to extend it by introducing a total

supportive tool for it. We performed it using a mapping between the concepts of Tropos to existing design elements of Jack.

We have used the extended Jack, EJack, to Analyze, Design, and implement several actual systems. EJack seems to support Tropos in all five phases, according to several systems implementation.

Another deficiency of Tropos is the lack of a testing/debugging tool. We are developing a testing tool , based on a two level model . We do the agent-test and the task-test using a tester agent .

Several open points still remain. We should consider concepts as reuse during all the activities during development process. In addition other important software engineering concepts, such as testing, debugging, deployment, and maintenance should be considered in Tropos.

Additionally, some important software engineering issues such as quality assurance, estimating guidelines, and supporting management decisions are not supported by this methodology.

In a higher stage, this work and similar ones may contribute another step towards developing the next generation of agent-oriented methodologies.

## References

- [1] P.Bresciani, P.Giorgini , F.Giunchiglia, J.Mylopoulos and A.Perini, "Towards an Agent Oriented approach to Software Engineering", *In the Workshop Dagli oggetti agli agenti: tendenze evolutive dei sistemi software*, Modena, Italy, 4-5 Sept 2001.
- [2] P. Bresciani, P. Giorgini, F. Giunchiglia , J. Mylopoulos and A.Perini, "Tropos: An Software Development Methodology", *In Journal of Autonomous Agents and Multi-Agent Systems*, Kluwer Academic Publishers, May 2004, pp. 203-236.
- [3] Busetta P., Ronnquist R., Hodgson A., and Lucas A., *Jack intelligent agents - components for intelligent agents in java*, Technical report, Agent Oriented Software Pty. Ltd., Melbourne, Australia, 1998.
- [4] J. Castro, M. Kolp, and J. Mylopoulos., "A requirements-driven development Methodology", *In Proceedings of the 13th International Conference on Advanced Information Systems Engineering, CAiSE'01*, Interlaken, Switzerland, June 2001, pp.108-123.
- [5] J. Castro, M. Kolp, and J. Mylopoulos. "Towards requirements-driven information systems engineering: The tropos project", *Information Systems*, Elsevier: Amsterdam, The Netherlands, 2002.

[6] M. Coburn, "Jack intelligent agents: User guide version 2.0", At <http://www.agent-software.com>, 2001.

[7] Message Web Site  
<http://www.eurescom.de/public/projects/p900-series/P907/P907.htm>

[8] A. Perini, P. Bresciani, F. Giunchiglia, P. Giorgini, and J. Mylopoulos, "A knowledge level software engineering methodology for agent oriented programming", *In proceedings of the 5th International Conference on Autonomous Agents, Agents'01*, Montreal, Canada, May 2001.

[9] M. Wooldridge and N.R. Jennings and D. Kinny, "A Methodology for Agent-Oriented Analysis and Design", *In Proceedings of the third international conference on*  
[11] L. Padgham, M. Winikoff, "Prometheus: A Methodology for Developing Intelligent Agents", *In Third International Workshop on Agent Oriented Software Engineering*, July 2002 .

[12] L. Padgham, M. Winikoff, "Prometheus: A Pragmatic Methodology for Engineering Intelligent Agents", *In Proceedings of OOPSLA 2002 WorkShop on Agent Oriented Methodologies* , pages 97-108 ,Seattle , November 2002 .

*Autonomous Agents and Multi-Agent Systems* , Seattle , WA , May 1999.

[10] M. Wooldridge and N.R. Jennings and D. Kinny, "The Gaia Methodology for Agent-Oriented Analysis and Design", *Autonomous Agents and Multi-Agent Systems* , 3(3) ,2000 .

[13] Yu, E., "Modeling Organizations for Information Systems Requirements Engineering", *In Proceedings of the First IEEE International Symposium on Requirements Engineering*, San Jose, USA, January 1993, pp. 34-41.