Extracting the Best Features of Two Tropos Approaches for the Efficient Design of MAS

Maria Jocelia Silva[1], Paulo Roberto Maciel[1], Rosa Cândida Pinto[1], Fernanda Alencar[2], Patrícia Tedesco[1], Jaelson Castro[1]

[1] Centro de Informática, Universidade Federal de Pernambuco, Caixa Postal 7851, Recife, PE, Brazil
{mjs2, prm, rccp, pcart, jbc}@cin.ufpe.br

[2] Departamento de Eletrônica e Sistemas, Universidade Federal de Pernambuco (UFPE), Caixa Postal 7146, Recife, PE, Brazil
fmra@ufpe.br

Abstract. Nowadays, there is an increasing demand for agent-oriented software development methodologies. Tropos is one of the most complete. However, researchers currently use two Tropos approaches. Some potential users feel difficulties in applying the methodology due to the differences in the sequence of activities and output artifacts generated by those two approaches. In this work we analyze both approaches of Tropos to identify the similarities and differences between them. As result, we select a sequence of activities and common output artifacts and extract the better points of each approach. The selected activities are then applied to the xaADB (a multiagent system that monitors DBMS) modeling to capture the fundamental concepts of the agents oriented development paradigm: agents, organization, communication, negotiation and coordination.

Keywords: agent oriented; methodology; multiagent systems; Tropos.

1 Introduction

The construction of a software product involves a series of often highly complex activities, which is developed from the point of view of some people that have the expertise on some part of the system. Such people can deeply know the product to be generated or can have only a partial comprehension of its general context, but they contribute with information and knowledge to attain the desired result. The use of methods and standards allow a correct documentation in all phases of the software life cycle. They ensure that all the information collected during the software development process can be identified, analyzed correctly and justified by its contributors.

A methodology is a collection of methods covering and connecting different stages in a process. Tropos [5, 6] is a software development methodology used for modeling of multiagent systems. It borrows the abstractions and concepts from organizational and social disciplines to understand, model, reason, analyze and design Multi-Agent

System (MAS) [19]. In fact these approaches provide a more flexible, higher-level set of constructs to deal with a world operating more on social principles than on mechanistic rules [20]. Currently, there are two versions of Tropos [5, 6]. There are differences in the sequence of activities and output artifacts generated.

The aim of this paper is analyses the main versions of Tropos methodologies [5,6]. At first, our goal is to develop the case study of multiagent software, the xaADB multiagent system. So we have to choice one of the versions. Thus we decided to identify the similarities and differences between them; to propose a hybrid sequence of activities and commons output artifacts of these approaches; and finally, to apply the selected sequence to the case study.

This paper is structured as follows. Section 2 presents the related works. Section 3 presents the two Tropos' approaches. Section 4 introduces the selection of the commons activities. The case study used to evaluate the use of the methodology is showed in section 5. Section 6 presents the results of the application of the selected activities in the case study. Finally section 7 presents the conclusions and the future works.

2 Related work

Sturn proposes in [17] an evaluation method for agent-oriented methodologies using a feature-based analysis, a survey, and a structured analysis. He compares Gaia MaSe, Tropos and OPM/MAs methodologies. Dam [10] introduces an evaluation framework for performing a systematic and comprehensive evaluation of several agent-oriented methodologies. Cernuzzi and Rossi in [8] propose a framework evaluate of the agent-oriented analysis and design methodologies considering qualitative evaluation criteria employing quantitative methods. They claim that their framework may be used by agent-based systems designer as well as for authors of agent-oriented methodologies. All of these works compare different methodologies. Our work presents a study of different versions of the same methodology.

Multiple studies have been proposed for increase the modeling agent-oriented systems. Cernuzzi and Zambonelli [9] introduce AUML [1] as notation into Gaia Methodology [21] for capture a very rich and expressive way the agent's interaction protocols. Bernon [2] tries joining the ADELFE, Gaia and PASSI methodology for unified their meta-models showing the interoperability between agent-oriented methodologies. Juan [14] introduces a conceptual framework for creating and reusing modular methodologies, their framework can modularize agent-oriented methodologies. None of these works presents a clear process to development agent-oriented software.

Our proposal outlines a guideline to help the software engineer to development agent-oriented systems. In our work we join the two versions selecting what have better in each one for apply in a case study of multiagent software.

3 The Two Tropos Approaches

Tropos was proposed in 2001 [7] to develop a framework for building agent software systems. This project group authors from various universities in Brazil, Canada, Belgium, Germany and Italy. Tropos adopts the concepts offered by *i** [20], a modeling framework based on concepts such as actor (actors can be agents, positions or roles), and social dependencies among actors (goal, softgoal, task and resource). Tropos spans four phases:

- Early requirements, concerned with the understanding of a problem by studying an organizational setting.
- Late requirements, where the system-to-be is described within its operational environment, along with relevant functions and qualities.
- Architectural design, where the system's global architecture is defined in terms of subsystems, interconnected through data, control and other dependencies.
- Detailed design, where behavior of each architectural component is defined in further detail.

The initial ideas of Tropos [7] were evolved in what we call Tropos'01 [6], and later in what we call Tropos'04, whose first studies were presented in [4], and soon afterwards in [5]. In this paper we considerer the approaches presented in [6] (the Toronto's version) and [5] (the Italy's version).

Based in [18], which propose a generic formalization of a spiral software development process for MAS, we analyzed the two Tropos versions. We show the differences between Tropos'01 and Tropos'04 identifying the activities and output artifacts.

By the analyses of the Tropos'01, we identify that the activities are textually expressed. This makes hard the comprehension and increases the analyst's work. Therefore, we organize, In the Table 1, the Tropos'01 activities with its occurrence sequence and output artifacts. We organize the activities as four phases of the methodology, labeled as follow: the two first letters relate the phases of the methodology (e.g., ER-Early Requirements, LR-Late Requirements, AD-Architectural design, DD-Detailed Design); the two next letters relate to versions of Tropos (e.g. 01-Toronto); and, finally the sequential number of the activity. Thus, ER.01.1 means the early requirements phase (ER) of the Tropos'01 (01) and the activity one (1). The artifacts are codified sequentially (A1..An). The activity ER.01.1 has A1 (Stakeholder List) as its output artifact.

By the same way, we analyzed the Tropos'04 [5], and proposed the Table 2. Note that this table has three columns instead two as in Table 1. The activities of the all phases are grouped in modeling stages (actor modeling, dependency modeling, goal modeling, plan modeling and capability modeling). We group the two first modeling stages (Actor Modeling and Dependency Modeling), in the Requirements phases (Early and Late), because the identification of actors and its dependencies are done simultaneous in this phases. We join these modeling stages in only one called Actor and Dependency Modeling. Besides this, the activities labels are changed in the field related with the Tropos' version (e.g. ER.04.01, means the early requirements phase (ER) of the Tropos'04 (04) and the activity one (1) and belong to the Actor and Dependency Modeling stage).

Altogether, the activities of both versions are similar in early requirements and late requirements. The architectural design and detailed design have some differences; in general, the sequence of activities is different to attain similar artifacts and some different artifacts (e.g., capability diagrams in the Tropos'04 and the UML class diagram in the Tropos'01.) are found. The next section discusses this similarities and differences and proposes a hybrid sequence of activity that will be applied in the case study.

Table 1. Tropos'01 Activities and output artifacts.

Phase / Activities	**Output Artifacts**
Early Requirements	
ER.01.1. Identify the stakeholders who will be represented as (social) actors who depend on each other for goals to be achieved, task to be performed and resource to be furnished.	A1 - Stakeholder List
ER.01.2. Produce a strategic dependency model (SD) for describing the network of relationships among actors. A strategic dependency model is a graph involving actors who have strategic dependencies among each other.	A2 - Strategic Dependency Model
ER.01.3. Once the relevant stakeholders and their goals have been identified, a strategic rationale model determines through a means-ends analysis how these goals (including softgoals) can actually be fulfilled through the contributions of other actors.	A3 - Strategic Rationale Model
Late Requirements	
LR.01.1. Identify the system 'to-be' who will be represented as one or more actors who contribute to the fulfillment of stakeholder goals.	A4 - Strategic Dependency Model
LR.01.2. Integrate the actors representing the system's environment and the system's functional and non-functional requirements in the Strategic Dependency Model.	A5 - Strategic Rationale Model
LR.01.3. The system is decomposed into several sub-actors which take on some of these responsibilities. This decomposition and responsibility assignment is realized using the same kind of means-ends analysis along with the strategic rationale analysis.	A6 - Strategic Rationale Model
Architectural Design	
AD.01.1. The first task during architectural design is to select among alternative architectural styles (e.g. hierarchical contracting, pyramid, joint-venture etc) [15][11] using as criteria the desired qualities identified earlier.	A7 - Selected Architectural Style
AD.01.2. Include new actors in current models and assign system responsibilities to included actors, based on the selected architectural style.	A8 - Strategic Dependency Model
AD.01.3. A further step in the architectural design consists in defining how the goals assigned to each actor are fulfilled by agents with respect to social patterns [12] (e.g. matchmaker, broker, mediator, wrapper, etc). For this end, designers can be guided by a catalogue of agent patterns in which a set of pre-defined solutions are available in Tropos, social patterns are used for solving a specific goal defined at the architectural level through the identification of organizational styles and relevant quality attributes (softgoals) as discussed previously.	A9 - Social patterns, Strategic Dependency Model
AD.01.4. A detailed analysis of each social pattern allows defining a set of capabilities associated with the agents involved. A capability states that an actor is able to act in order to achieve a given goal. In particular, for each capability the actor has a set of plans that may apply in different situations. A plan describes the sequence of actions to perform and the conditions under which the plan is applicable.	A10 - Social Pattern Capabilities List
AD.01.5. Capabilities are collected in a catalogue and associated to the pattern. This allows defining the actors' role and capabilities suitable for a particular domain.	A11 - Social Pattern Capabilities List
AD.01.6. The goal is decomposed into different sub goals and solved with a combination of patterns	A12 - Strategic Rationale Model.
Detailed Design	
DD.01.1. Model a UML class diagram [3] focusing on that actor that will be implemented. The target implementation model is the BDI model [16], an agent model whose main concepts are Beliefs, Desires and Intentions.	A13 - UML class diagram.
DD.01.2. Model events exchanged in the system, using extended AUML [1] sequence diagrams and collaborations diagrams.	A14 - AUML sequence diagram.
DD.01.3. At the lowest level, use plan diagrams, to specify the internal processing of atomic actors. Each identified plan is specified as a plan diagram. The plan graph is a UML state transition diagram.	A15 - Plan Diagram

Table 2. Tropos'04 activities and output artifacts.

Phases/ Modeling	**Activities**	**Output Artifacts**
Early Requirements		
Actor and Dependency Modeling	ER.04.1. Identify and analyze the stakeholders and their intentions. Stakeholders are modeled as social actors who depend on one another for goals to be achieved, plans to be performed, and resources to be furnished. Intentions are modeled as goals.	A1 - Stakeholder List
Goal Modeling	ER.04.2. Through a goal-oriented analysis, goals are decomposed into finer goals, which eventually can support evaluation of alternatives. ER.04.3. The rationale of each goal relative to the stakeholder who is responsible for its fulfillment has to be analyzed. Goal decomposition can be closed through a means-end analysis aimed at identifying plans, resources and softgoals that provide means for achieving the goal. ER.04.4. Inside the actor diagram, softgoal analysis is performed identifying the goals that contribute positively or negatively to the softgoal.	A2 - Actor Diagram A3 - Goal Diagram
Late Requirements		
Actor and Dependency Modeling	LR.04.1. The system-to-be is represented as one actor which has a number of dependencies with the other actors of the organization. These dependencies define the system's functional and non-functional requirements.	A4 - Actor Diagram for system actor
Goal Modeling	LR.04.2. These goals are then analyzed from the point of view of the actor. LR.04.3. Softgoal contributions can be identified applying the same kind of analysis described by the goal diagram.	A5 - Goal Diagram for system actors
Plan Modeling	LR.04.4. Decompose the plans to complement the goal modeling.	
Dependency Modeling	LR.04.5. Some dependencies in the actor diagram must be revised upon the introduction of the system actor.	A4 - Actor Diagram
Architectural Design		
Actor Modeling	AD.04.1. Inclusion of new actors and delegation of sub-goals to sub-actors upon goal analysis of system's goals.	A6 - Actor Diagram for Architecture
Dependency Modeling	AD.04.2. Inclusion of new actors according to the choice of a specific architectural style [11]. AD.04.3. Inclusion of actors contributing positively to the fulfillment of some specific functional and non-functional requirement.	A7 - Extended Actor Diagram
Capability Modeling	AD.04.4. This step consists in the identification of the capabilities needed by the actors to fulfill their goals and plans. Capabilities are not derived automatically but they can be easily identified by analyzing the extended actor diagram.	A8 - Actor Capabilities List
	AD.04.5. The last step consists of defining a set of agent types and assigning each of them one or more different capabilities (agent assignment). Tropos suggests a set of pre-defined patterns recurrent in multi-agent literature that can help the designer [12].	A9 - Agent Types Capabilities List
Detailed Design		
Capability Modeling	DD.04.1. The UML activity diagram [3] allows us to model a capability (or a set of correlated capabilities) from the point of view of a specific agent.	A10 - Capability Diagrams
	DD.04.2. Each plan node of a capability diagram can be further specified by UML activity diagrams.	A11 - Plan Diagrams
	DD.04.3. Here AUML sequence diagrams [1] can be exploited. In AUML sequence diagrams, agents correspond to objects, whose lifeline is independent from the specific interaction to be modeled communication acts between agents correspond to asynchronous message arcs.	A12 - Agent Interaction Diagrams.

4 The Selected Sequence of Activities

The two Tropos versions [5, 6] were used to select the better sequence of activities that allow the methodology users apply the best features of each approach. In this section, these versions are analyzed and for each methodology phase a sequence of activities are selected and presented in Table 3. Below, these analyses are split in phase.

Early and Late requirements: Tropos'01 and Tropos'04 generate similar output artifacts: Stakeholders List, Actor Diagram with Strategic Dependency Model and Goal Diagram with Strategic Rationale Model. The difference between the two lies in the proposed sequence of activities. In Tropos'01 the goals are fulfilled following the original proposal of the i* [20]. Tropos'04 besides the means-end analysis offers more types of decompositions (e.g., AND-decomposition and OR-decomposition). Thus, Tropos'04 is a clearer method for creating the Actor diagram and Goal diagram. Furthermore, models are constructed by analyzing each intentional element and applying the follow modeling stages: actor modeling, dependency modeling, goal modeling and plan modeling. This allows create diagrams easier to understand. Therefore, we selected the Tropos'04 sequence of activities to the both phases (Table 3).

Table 3. Selected sequence of activities of Tropos

Selected Activities	Artifacts
Early Requirements	
ER.04.1 - Identify the stakeholders and their intentions. ER.04.2 – Decompose goals through a goal-oriented analysis. ER.04.3 - Analyze the rationale of each goal. ER.04.4 - Identify goals that contribute positively or negatively to softgoal.	A1 - Stakeholder List A2 - Actor Diagram A3 - Goal Diagram
Late Requirements	
LR.04.1 - Identify the system 'to-be'. LR.04.2 – Analyze the goals from the point of view of the actor. LR.04.3 – Identify softgoal contributions. LR.04.4 – Decompose the plans. LR.04.5 – Revise dependencies in the actor diagram.	A4 - Actor Diagram A5 - Goal Diagram
Architectural Design	
AD.04.1 - Include new actors and delegation of sub-goals to them. AD.01.1 - Select among alternative architectural styles. AD.01.2 - Include new actors based on the selected architectural style. AD.01.3 - Define agents with respect to social patterns. AD.01.4 - Define a set of capabilities associated with the agents involved in the pattern. AD.04.5 - Define a set of agent types (social patterns).	A6 - Actor Diagram for System Architecture A7 - Selected Architectural Styles A8 - Actor Diagram A9 - Goal Diagram A10 - Actor Capabilities List A11 - Selected social patterns
Detailed Design	
DD.01.1 - Model a UML class diagram. DD.04.1 - Model a capability. DD.04.2 – Specify each plan node of a capability diagram by UML activity diagrams. DD.01.2 - Model events using extended AUML sequence diagrams and collaborations diagrams.	A12 - UML class diagram A13 - Capability Diagram A14 - Plan Diagrams A15 - Agent Interaction Diagrams

Architectural design: the global architecture of the system is defined in term of subsystems represented by new actors. Tropos'01 first defines the architectural styles. From the selected style, new actors are inserted and its system goals are attributed. The goals are filled with respect to the social patterns by analyzing their capacities. The Tropos'04 defines the actors from the analysis of the goals, selects the style architectural and includes new actors who contribute to fill the functional and non-functional requirements. It identifies the capacities of for each actor and classifies the types of agents who fulfill the goals using the social patterns. The results of both approaches are the same, but we choose a sequence that emphasizes system goals instead of the architectural styles and social patterns. The selected activities (Table 3) are: AD.04.1 from Tropos'04; AD.01.1, AD.01.2, AD.01.3, AD.01.4 from Tropos'01, and AD.04.5, again from the Tropos'04). These activities improve the using of the methodology in our study case.

Detailed design: following the same process logic of the later phase, the activities chosen for this phase also come from both Tropos'01 and Tropos'04. The first activity is to implement a class diagram of each agent (DD.01.1). After this, the diagram of capacities (DD.04.1) and each plan (DD.04.2) that composes this diagram are further detailed. Finally, the protocols of communication of the agents are specified through the sequence diagram from AUML (DD.01.2). It is observed that the sequence of activities is the same, with addition of class model, that has the advantage to express a more complete agents modeling.

The summary of the sequence of activities selected for our case study is showed in Table 3. As explained, these activities are collected from Tropos'01 (see Table 1) and Tropos'04 (see Table 2). The codification of the labels is the same described above. Table 3 shows that all activities in the Early and Late Requirements were selected from Tropos'04. In architectural design some activities was selected from Tropos'01 (AD.01.1, AD.01.2, AD.01.3, AD.01.4) and Tropos'04 (AD.04.1, AD.04.5). In Detailed design also was selected activities from Tropos'01 (DD.01.2, DD.01.1) and Tropos'04 (DD.04.1, DD.04.2).

The next section presents concepts of the system xaADB, which is the case study of this work.

5 Case Study

To get a deeper understanding of the Tropos methodology, a system architectural specification was developed. In order to make the case study as complete as possible, we have modeled the XAADB (eXternal Architecture for Autonomous Database administration), a multi-agent system that aims at providing autonomy to Database Management Systems (DBMS). This system is presented as an architectural framework, for autonomous fault resolution components development for traditional DBMS. The XAADB is being developed as part of a Master's research at the Informatics Center, Federal University of Pernambuco, Brazil.

Acting as an electronic Database Administrator, the xaADB involves catalogued fault resolution, fail alerts, bad settings alerts and performance trouble resolution.

Some implementation requirements also should be followed (e.g. using of intelligent agents, software developed as an external layer from DBMS software, using of machine learning techniques as well as closely following the autonomic computing principles [13]). This case study has a special characteristic of being predominantly a computational system (not only an information system) which brings the specification of non-functional requirements to the forefront. Database administration is a well known activity in informatics, thus making it easier to evaluate the case study with respect to the concepts described in this paper.

6 Applying the Selected Sequence

We apply the Tropos sequence suggested in section 3 on the system xaADB. The models were drawn to represent xaADB system components.

5.1 Early Requirements: in this phase, stakeholders and their intentions are modeled. The xaADB is a "computational" system that aims at helping a data base administrator to solve specific problems in DBMS. This phase justify the need of this system to the DBA and the organization where the DBA is working for. The artifacts produced by the activities below represent who are the stakeholders and their needs.

ER.04.1 - Identify the stakeholders and their intentions: There are two main stakeholders in the system xaADB. The actor Client is the organization that decides on about the software acquisition. The actor DBA is the database administrator that works in the organization. The stakeholders and the list of their needs are showed in table 4.

Table 4. Stakeholder List

	Stakeholder	Objectives
1	DBA	1. To liberate him/herself of routine tasks; 2. To have resources allow him/her to improve the work quality; 3. To use more time for strategical tasks.
2	Client / Company	1. More efficient administration of data to a lesser cost; 2. To improve the allocation of human resources in the company. 3. To guarantee the continuity of services.

Figure 1 show the stakeholders needs, in a graphical notation using i* model [20]. Actor Client has two main goals, which are obtain the "Data management improved" and the "Cost reduced". It depends on actor DBA to fulfill the goal "Database management performed". The Client expects "Efficiency in data management provided" and the "Service continuity insured". The DBA expects the Client to achieve better work conditions. To do his/her job, the DBA uses the services provided by the Operational System (OS) and Database management system (DBMS). They are represented as system actors.

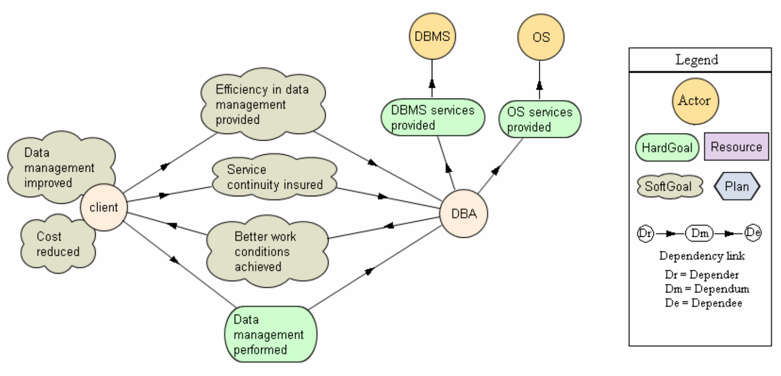


Fig. 1. Artifact actor diagram or Strategic dependency model

ER.04.2 – Decompose goals through a goal-oriented analysis: The Client has as main goals the "*Data management improved*" and "*Betters work condition achieved*". DBA has a one goal which is "Data management performed". In this case, there are no goal decompositions.

ER.04.3 - Analyze the rationale of each goal: Client can acquire a new solution that contributes positively to obtain his goal. To acquire a new solution, the Client performs the plan "Evaluate Solutions". S/He has three ways to evaluate. S/He can "Develop a tool" or "Buy a tool" or "Use a free tool". The actor DBA has the goal "Database management performed". S/He can perform two plans to obtain this goal. S/He either continues the "Use of traditional administration" or "Automate task using a SMA". The multiagent system which realizes this task is the xaADB system. Figure 2 shows the goal diagram to DBA and Client actor.

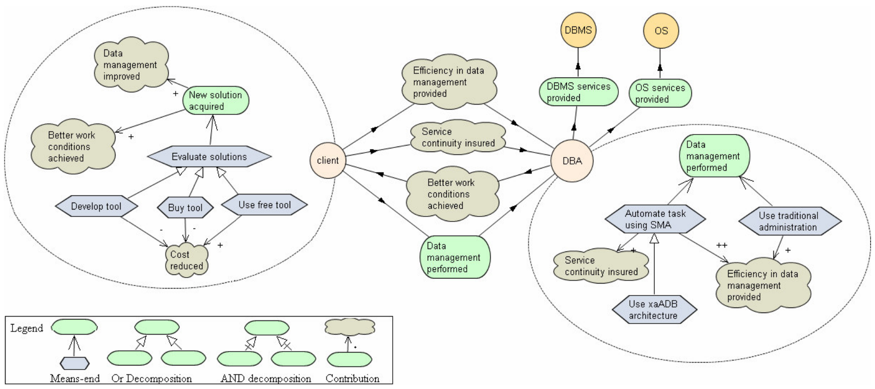


Fig. 2. Artifact goal diagram or Strategic rationale model (SR) for the actor Client.

ER.04.4 - Identify the goals that contribute positively or negatively to the softgoal: Among the Client evaluated alternatives to acquire a new solution the plan "*Use a free tool*" contributes positively to obtain "*cost reduced*". The other options contribute negatively. Use a tool with requirements of a SMA contributes positively to achieve the Client soft goal "*Efficiency in Data management provided*". Traditional administration also contributes to obtain this soft goal, but with a minor pound of efficiency. Plan "*Use automated task using a SMA*" contributes positively to softgoal "*Service continuity insured*". The figure 2 shows this analysis.

5.2 Late Requirements: This phase focus on actor's dependencies on system and analyze the goals to identify non-functional and functional requirements. The main goal that generates the software is analyzed and decomposed in sub-goals and plans. These represent the software functionality. The following activities compose the Late Requirement phase in xaADB software.

LR.04.1 - Identify the 'system-to-be': in this activity, the xaADB system is introduced in the model and its dependencies on others actors are identified. The actor DBA depends on xaADB system to fulfill the goal "Autonomous fault resolution performed". The expected quality attributes from xaADB are represented as softgoals. These are "High performance", "Usable system", "Trustful system", "Supportable system" and "Secure system". xaADB uses the services provided by Operational system and the DBMS. Figure 3 shows the Actor diagram with xaADB system.

LR.04.2 – Analyze the goals from the actor's point of view: The system main goal is "Autonomous fault resolution performed". It is decomposed in sub-goals that represent different classes of problems that will be solved by the system. The system monitors the DBMS and operational system status. When a fault is detected, a resolution method must be applied to solve the detected problem. Figure 4 shows the decomposed sub-goals in the area marked with label LR.04.2.

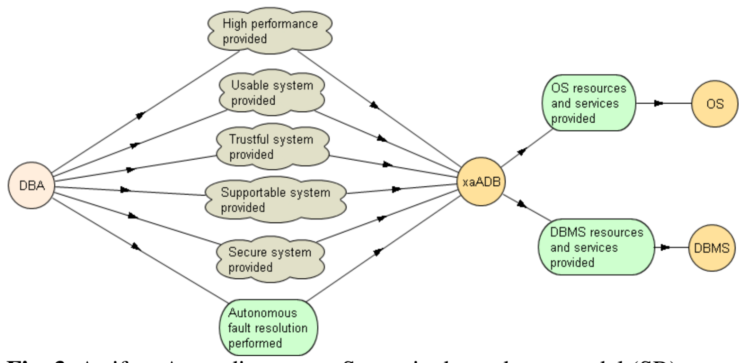
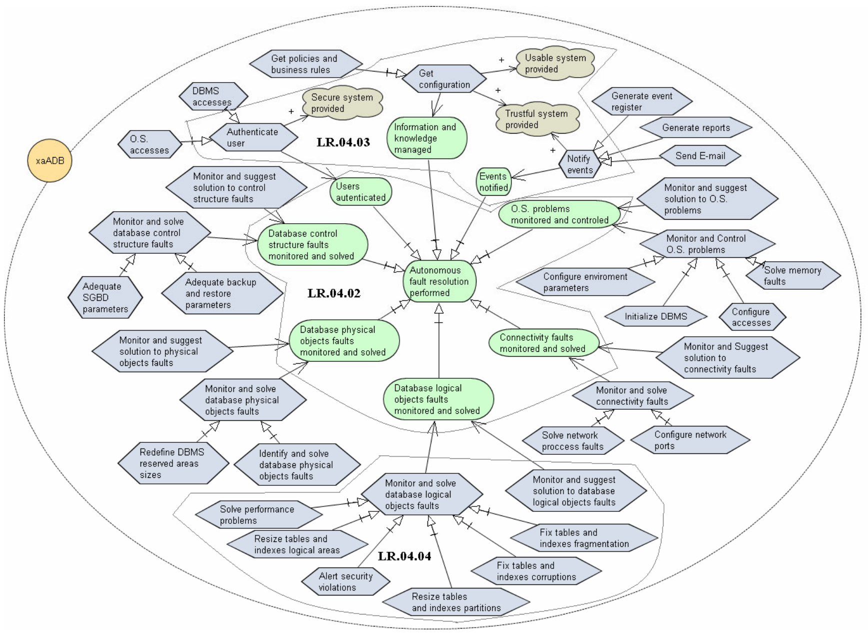

High performance provided
Usable system provided
Trustful system provided
Supportable system provided
Secure system provided
Autonomous fault resolution performed
DBA
xaADB
OS resources and services provided
OS
DBMS resources and services provided
DBMS

Fig. 3. Artifact Actor diagram or Strategic dependence model (SR).

LR.04.3 – Identify softgoal contributions: To satisfy the DBA quality expectation, new goals are included in the model. “Users authenticated” contribute to achieve the softgoal “Secure System provided”. “Information and knowledge managed” and “Events notified” are included to fulfill the softgoal “Usable system provided” and “Trustful system provided”. Figure 4 shows the softgoal contribution delimited by the area labeled LR.04.3**.**

LR.04.4 – Decompose the plans: The analysis of xaADB system is finished by the identification of plans that will be executed to achieve the goals. To demonstrate this activity, a means-end analysis of the goal “Database logical objects faults monitored and solved” was made. The system monitors the DBMS status. If a failure is detected, it chooses between correcting the failure or suggesting a solution to the DBA. The failure resolution can be decomposed in many sub-plans to solve the detected different problems. The area marked with the label LR.04.4 in the figure 4, shows the plan decomposition. Each goal is analyzed during the late requirements. The final result is the complete goal diagram shown in figure 4.

LR.04.5 – Revise dependencies in the actor diagram: when the analysis of goals and plans is complete, new dependencies between the xaADB and others actors can be identified. For the sake of space, this analysis will not be shown here.



xaADB
Get policies and business rules
Get configuration
Usable system provided
DBMS accesses
Secure system provided
Trustful system provided
Generate event register
O.S. accesses
Authenticate user
LR.04.03
Information and knowledge managed
Generate reports
Notify events
Send E-mail
Monitor and suggest solution to control structure faults
Users autenticated
Events notified
Monitor and suggest solution to O.S. problems
Monitor and solve database control structure faults
O.S. problems monitored and controled
Database control structure faults monitored and solved
Monitor and Control O.S. problems
Autonomous fault resolution performed
Configure enviroment parameters
Solve memory faults
Adequate SGBD parameters
Adequate backup and restore parameters
LR.04.02
Initialize DBMS
Configure accesses
Monitor and suggest solution to physical objects faults
Database physical objects faults monitored and solved
Connectivity faults monitored and solved
Monitor and Suggest solution to connectivity faults
Monitor and solve database physical objects faults
Database logical objects faults monitored and solved
Monitor and solve connectivity faults
Solve network proccess faults
Configure network ports
Redefine DBMS reserved areas sizes
Identify and solve database physical objects faults
Monitor and solve database logical objects faults
Monitor and suggest solution to database logical objects faults
Solve performance problems
Resize tables and indexes logical areas
Fix tables and indexes fragmentation
LR.04.04
Fix tables and indexes corruptions
Alert security violations
Resize tables and indexes partitions
+
+
+
+

Fig. 4. Artifact goal diagram or Strategic rationale model (SR) for the actor xaADB

5.3 Architectural design: In this phase, the actor xaADB is analyzed and new actors are identified. The goals are delegated to new actors. First, roles assumed by the actor are analyzed; next roles are broken in agents with special capabilities.

AD.04.1 - Include new actors and delegation of sub-goals to them: The system goals were grouped in "Monitor and solve faults", "Manage information and knowledge" and "Register and notify events". Actors "Fault solver", "Information and knowledge manager" and "Report Manager" were created to achieve these goals, respectively. "Fault Solver" is a role which can be played by others actors, depending on the fault to be solved. Actors "Connectivity Manager" and "O.S. Manager" are agents who monitor and solve connectivity and O.S. failures. Figure 5 shows the actor diagram for system architecture.

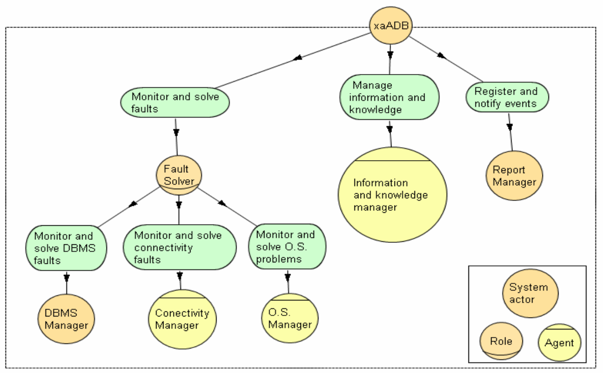

xaADB
Monitor and solve faults
Manage information and knowledge
Register and notify events
Fault Solver
Information and knowledge manager
Report Manager
Monitor and solve DBMS faults
Monitor and solve connectivity faults
Monitor and solve O.S. problems
DBMS Manager
Conectivity Manager
O.S. Manager
System actor
Role
Agent

Fig. 5. Artifact Actor Diagram for System Architecture

AD.01.1 - Select among alternative architectural styles: Tropos uses the organizational styles proposed in [11][15]:pyramid, joint venture, structure in 5 and more. An architectural organization is defined coming from the system quality attributes, presented as softgoals. Using these criterions to select organizational structure, two possibilities were found: Joint Venture and Pyramid. Joint-venture organization style is more decentralized, which permits greater autonomy to local actors. So Joint Venture style was chosen for xaADB system.

AD.01.2 - Include new actors based on the selected architectural style: Following the Joint venture style, the actor Coordinator was inserted into model. Actor "Fault Solver" delegate global plans execution control to actor Coordinator. Actor "Information and knowledge manager" provide information to other actors. Both "Coordinator" and "Fault Solver" can send information to system user about performed actions to "Report manager" actor. Figure 6 shows the new actors in selected architectural style.

AD.01.3 - Define agents with respect to social patterns: After select the architectural style, the plans and objectives are analyzed and new actors are inserted from agents' social patterns mappings [12]. Due to space limitation, we will end the xaADB analysis at this point. In future work, the analysis of social patterns' use and detailed design phase will be presented.

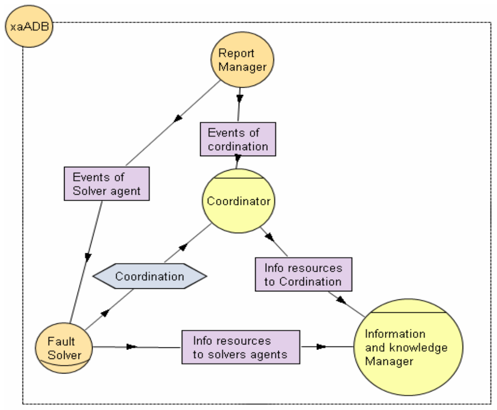


Fig. 6. Selected Architectural Styles

7 Conclusions and future work

This paper focuses on the identification of activities that capture the best points of two approaches of the Tropos development methodology. To achieve this, the four Tropos' phases were analyzed for the both approaches. As a result, a common sequence of activities and output artifacts was generated. The selected activities were applied to a multiagent system that monitors distributed DBMS: the xaADB system. In this work, some activities of the phases Early Requirements, Late Requirements and Architectural Design were modeled. The objective was to present how the selected activities' sequence could help us to use the Tropos methodology.

For future work, we plan to analyze deeply the detailed design phase and model the xaADB system. For this, we will compare Tropos with GAIA [21]. The goal will be to evaluate which methodology is more adequate to detailed design. Tropos is more complete for Early Requirements and Late Requirements. We also plan to develop a process to guide the use of Tropos to Multi-agent system's modeling and allow traceability in the methodology phases.

References

1. Bauer, B., Muller, J. and Odell, J.: Agent UML: A formalism for specifying multiagent interaction. In Proc. of the 1st Int. Workshop on Agent-Oriented Software Engineering, AOSE'00, pages 91–104, Limerick, Ireland (2001).
2. Bernon, C., Cossentino, M., Gleizes, M., Turci, P., and Zambonelli, F. "A Study of Some Multi-Agent Meta-Models", in Odell, J., Giorgini, P., and Muller, J., editors, *Agent-oriented Software Engineering V, AOSE 2004*, volume 3382 of LNCS, pages 62–77. Springer-Verlag, Berlin, Heidelberg. (2004).
3. Booch, G., Rumbaugh, J. and Jacobson, I.: The Unified Modeling Language: User Guide. Addison-Wesley (1999).

4. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: Towards an Agent Oriented approach to Software Engineering. In the Workshop Dagli oggetti agli agenti: tendenze evolutive dei sistemi software (2001).
5. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J. and Perini, A.: Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems, 8(3):203–236, (2004).
6. Castro, J., Kolp, M. and Mylopoulos, J.: Towards Requirements-Driven Information Systems Engineering: The Tropos Project. Information Systems Journal, Elsevier, Vol 27: 365-89 (2002).
7. Castro, J., Kolp ,M., Mylopoulos, J.: UML for Agent-Oriented Software Development: the Tropos Proposal. In International Conference on the Unified Modeling Language (2001).
8. Cernuzzi, L. and Rossi, G. “On the Evaluation of Agent Oriented Methodologies”, Proceedings of the Workshop on Agent Oriented Methodology (OOPSLA’02). Pages: 21-32. COTAR, (2002).
9. Cernuzzi , L. and Zambonelli, F. “Experiencing AUML in the Gaia Methodology “, in 5th International Conference on Enterprise Information Systems. Angers - France April 2003.
10. Dam, K. H. and Winikoff, M. “Comparing Agent-oriented Methodologies”, in Giorgini, P., Henderson-Sellers, B., and Winikoff, M., editors, Agent-Oriented Information Systems: 5th International Bi-Conference Workshop, volume 3030 of LNAI, pages 78–93. Springer-Verlag, Berlin, Germany. (2003).
11. Fuxman, A., Giorgini, P., Kolp, M. and Mylopoulos, J.: Information systems as social structures. In Proc. of the 2nd Int. Conf. on Formal Ontologies for Information Systems, FOIS’01, Ogunquit, USA (2001).
12. Hayden, S., Carrick, C. and Yang, Q.: Architectural design patterns for multiagent coordination. In Proc. of the 3rd Int. Conf. on Autonomous Agents, Agents’99, Seattle (1999).
13. Horn, P.: Autonomic Computing: IBM's Perspective on The State of Information Technology - IBM Corporation. <http://www.research.ibm.com/autonomic>, last access Jan 2007.
14. Juan, T., Sterling, L., Martelli, M., and Mascardi, V. “Customizing AOSE Methodologies by Reusing AOSE Features”, in Proceedings of the 2nd Iinternational Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS’03), pages 113–120, New York, USA. ACM Press. (2003).
15. Kolp, M., Castro, J. and Mylopoulos, J.: A social organization perspective on software architectures. In Proc. of the 1st Int. Workshop From Software Requirements to Architectures, STRAW’01, pages 5–12, Toronto, Canada (2001).
16. Rao, A., and Georgeff, M.: Decision procedures for BDIlogics. Journal of Logic and Computation 8(3):293–344 (1998).
17. Sturm A., Dori, D. & Shehory O.: Comparative Evaluation of Agent-Oriented Methodologies, in Methodologies and Software Engineering for Agent Systems, Kluwer, pp. 127-149 (2004).
18. Wautelet Y., Kolp M. and Achbany Y.: S-Tropos, An Iterative SPEM-Centric Software Project Management Process, Working Paper IAG (2005).
19. Wooldridge, M.: An Introduction to MultiAgent Systems. John Wiley and sons, LTD, Chichester, England (2002).
20. Yu, E.: Modelling Strategic Relationships for Process Reengineering. PhD thesis, University of Toronto, Department of Computer Science (1995).
21. Zambonelli, F., Jennings, N. R., Wooldridge, M.: Developing Multiagent Systems: The Gaia Methodology. ACM Transactions on Software Engineering Methodology, v. 12, n. 3, p. 317-370 (2003).