Proceedings of the Int. Conf. on Computer and Communication Engineering, ICCCE'06
Vol. I, 9-11 May 2006, Kuala Lumpur, Malaysia

EJACK: AN Extended Developing Tool for Tropos

M.KazemiFard[1], A.Fatemi[2], N.NamatBakhsh[3]
Department of Computer Engineering, University of Isfahan, Isfahan, Iran
[1]KazemiFard@eng.ui.ac.ir, [2]Afsaneh_Fatemi@hotmail.com, [3]Nemat@eng.ui.ac.ir

Abstract

Numerous methodologies for developing agent based systems have been proposed in the literature. However, their application is still limited due to their lack of sufficient documents, tools and maturity. Tropos is one of the most accurate and complete agent oriented methodologies that have been proposed in recent years. Several developing tools are designed to support Tropos, but almost non of them aren't able to support all phases of Tropos. One of these tools is JACK. JACK is a powerful tool for visual modeling of the architectural design and plans. It can support only the last three phases of Tropos: architectural design, detailed design, and implementation. In this paper, we proposed a method for mapping all concepts of Tropos to existing design elements of JACK. Then, we will be able to use the extended JACK, named EJACK, as a powerful developing tool for Tropos. We have used EJACK to Analyze, Design, and implement several actual systems. EJACK seems to support Tropos in all five phases, according to several systems implementation.

1. Introduction

Agent-oriented programming (AOP) is most often motivated by the need of open architectures that continuously change and evolve to accommodate new components and meet new requirements. More and more, software must operate on different platforms, without recompilation, and with minimal assumptions about its operating environment and its users. It must be robust, autonomous and proactive [1].

The key feature which makes it possible to implement systems with the above properties, is that, in this paradigm, programming is done at a very abstract level, more precisely, at the knowledge level. Thus, in AOP, we talk of mental states, of beliefs instead of machine states, of plans and actions, instead of programs, and so on.

Several agent-oriented software engineering methodologies, to cover AOP concepts, is proposed in literature [2].

Tropos is one of the most complete agent-oriented software development methodologies in which AI derived mentalistic notions such as actors, goals, soft goals, plans, resources, and intentional dependencies are used in all the phases of software development, from the first phases of early analysis down to the actual implementation [2]. A crucial role is given to the earlier analysis of requirements that precedes prescriptive requirements specification [3].

Tropos rests on five main phases for agent based systems development: early requirements analysis, late requirements analysis, architectural design, detailed design, and implementation. An incremental and iterative development process is adopted inside each phase and among different phases, in particular during early requirements analysis and late requirements analysis [1, 3, and 4].

One of the most deficiencies of Tropos is the lack of a complete tool to support all phases of it. JACK Intelligent Agents is an agent-oriented development environment built on top and fully integrated with Java [5]. It can support phases 3 to 5 of Tropos.

In this paper, we propose a mapping between Tropos concepts and JACK design elements, through which JACK can be used as a complete tool for Tropos. The extended tool is called EJACK[1].

The paper is structured as follows. In section 2, the Tropos methodology and its supporting tools are described. The JACK intelligent agent is introduced in Section 3, and the proposed mapping between Tropos concepts and its concepts, as the basis of EJACK, are then described. An e-Registering system, designed and implemented through EJACK, is presented in section 4 as a case study, while Section 5 summarizes the results of the paper and offers directions for future work.

[1] Extended JACK

ISBN 983-43090-0-7

2. Tropos

Tropos is a software development methodology that allows us to exploit all the flexibility provided by AOP. In a nutshell, the two novel features of Tropos are [3, 4]:

1. The notion of agent and related mentalistic notions are used in all software development phases, from early requirements analysis down to the actual implementation. These notions are founded on BDI[2] agent architectures.

2. A crucial role is given to early requirements analysis that precedes the prescriptive requirements specification of the system-to-be. This means that the methodology includes earlier phases of the software development process than those supported by other agent or object oriented software engineering methodologies.

The Tropos methodology is intended to support all the analysis and design activities from the very early phases of requirements engineering down to implementation, and organizes them into five main development phases [1, 3, 4, and 6]: early requirement analysis, late requirements analysis, architectural design, detailed design and implementation.

The main objective of the early requirement analysis in Tropos is the understanding of a problem by studying an existing organizational setting. During this phase, the requirement engineer models the stakeholders as actors and analyzes their intentions that are modeled as goals which, through a goal-oriented analysis, are then decomposed into finer goals that eventually can support evaluation of alternatives.

The late requirement analysis aims at specifying the system-to-be within its operating environment, along with relevant functions and qualities. The system is represented as an actor which has a number of dependencies with the actors already described during the early requirements phase. These dependencies define all functional and nonfunctional requirements for the system-to-be.

The main objective of the architectural design phase is the definition of the system's global architecture in terms of subsystems (actors), interconnected through data and control flows (dependencies).

The detailed design phase aims at specifying the agent capabilities and interactions. At this point, usually, the implementation platform has already been chosen and this can be taken into account in order to perform a detailed design that will map directly to the code.

[2] belief, desire, and intention

The implementation activity follows step by step the detailed design specification, according to the established mapping between the implementation platform constructs and the detailed design notions.

The four last phases are defined clearly in software engineering and are supported with various methodologies and tools. The first phase, early requirements analysis, is also defined in requirements engineering, but there is not enough experience on it. The key concepts of Tropos are showed in Table 1.

Table 1: The key concepts of Tropos

Concept	Description	Symbol
Actor	Model of Software Agents	
Soft Goal	Non Functional Goals	
Hard Goal	Functional Goals	
Plan	A Way of Doing Something	
Resource	a physical or an informational entity	
Dependency	The Relation Between Two Actors	

There is a lack of a complete tool to support all five phases of Tropos. Table 2 shows some of existing tools and phases they support [1].

Table 2: 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 2 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.

3. JACK and EJACK

JACK is a BDI 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 [5].

To support the programming of BDI agents, JACK offers five principal language constructs. These are *as* follows [7]*:*

– Agent: A JACK agent is used to define the behavior of an intelligent software agent. This includes the capabilities an agent has, the types of messages and events it responds to and the plans it uses to achieve its goals.

– Capability: A JACK capability can include plans, events, beliefs and other capabilities. An agent can be assigned a number of capabilities. Furthermore, a given capability can be assigned to different agents. JACK's capability notion provides a means to reuse.

– Belief: The JACK database amounts to a generalized relational database that describes a set of beliefs ascribed to an agent.

– Event: Internal and external events specified in the detailed design map to JACK's event construct. In JACK, an event describes a triggering condition for agent's actions.

– Plan: The plans contained in a capability specification resulting from a detailed design map to JACK plans. In JACK, a plan is a sequence of instructions the agent follows to try to achieve goals and deal with occurrences of events.

Tropos adopts the *i** modeling framework [8], which offers the notions of *actor*, *goal* and (actor) d*ependency*, 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 [9].

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

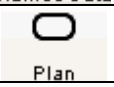
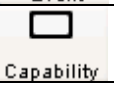
The diagrams produced in stages 1 and 2 by TAMO4E, should be converted to include JACK concepts, because of the difference between JACK and Tropos in key concepts. So, according to figure 1, we first convert Tropos concepts to BDI concepts and then change them to JACK concepts. This mapping, that is the basis of EJACK, is shown in table 3.

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

The student logins 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.

Figures 2, 3 and 4 show some of the designed diagrams, due to Tropos concepts. These diagrams can be generated in EJACK due to JACK development concepts. Figure 5 shows the architectural design diagram generating with EJACK.

Table 3: Mapping between Tropos and JACK concepts to construct EJACK

Tropos	JACK
Actor	
Resource	
Plan	
Soft Goal & Hard Goal	
Actor Capability	

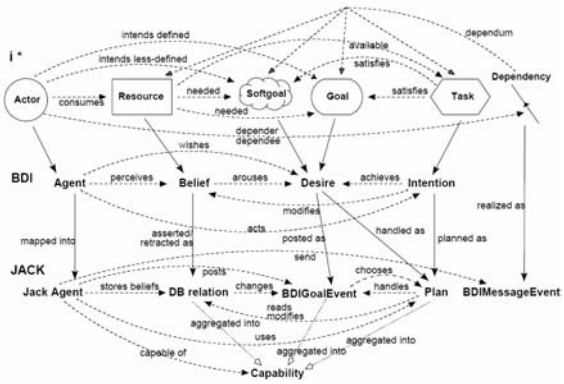


Figure 1: Converting Tropos concepts to JACK's, using BDI model concepts4. A Case Study

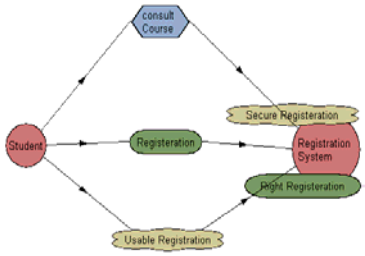


Figure 2: The early requirements diagram

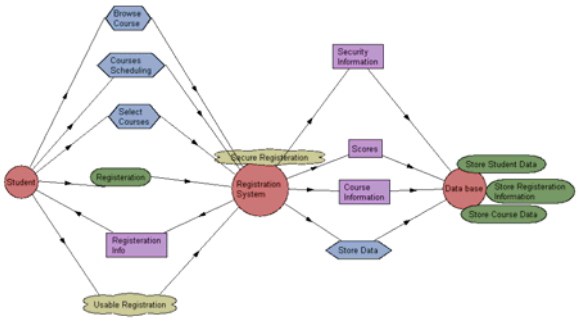


Figure 3: The late requirements diagram

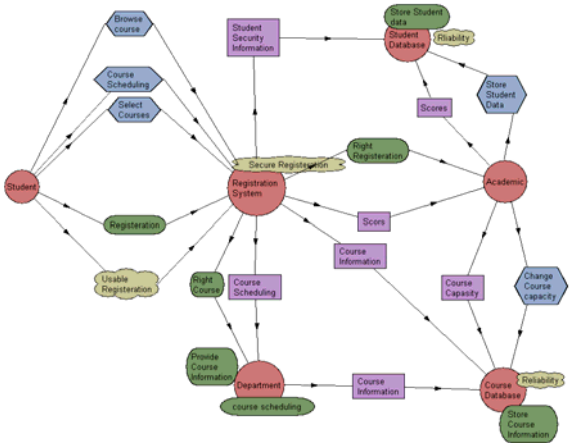


Figure 4: The architectural design diagram

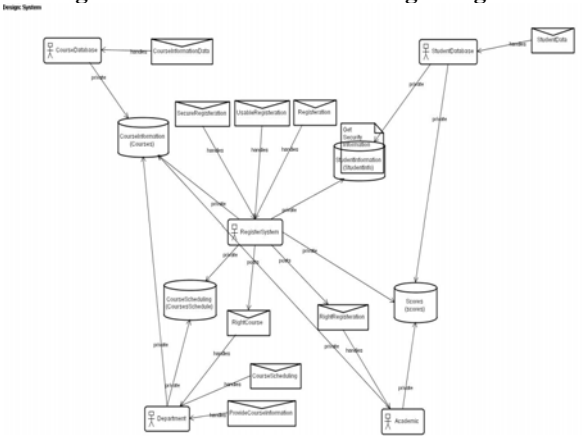


Figure 5: The architectural design diagram generated by EJACK

5. 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.

Tropos is one of the most accurate and complete agent oriented methodologies that covers software development from early requirement analysis to implementation. Lacking a proper tool to cover all five phases is one of the most essential deficiencies of Tropos.

In this paper, we proposed a method to use existing JACK development tool, as a complete tool for Tropos. We do this through mapping all concepts of Tropos to existing JACK concepts resulting EJACK development tool. EJACK seems to be an appropriate tool to support Tropos, duo to experimental results.

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, "Tropos: An Software Development Methodology*", In Journal of Autonomous Agents and Multi-Agent Systems*, Kluwer Academic Publishers, May 2004, pp. 203-236.

[2] K. H. Dam , Evaluating and Comparing Agent-Oriented Software Engineering Methodologies , Ms Thesis, School of Computer Science and Information Technology, RMIT university, Australia, 2003.

[3] 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.

[4] 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.

[5] P. Busetta, R. Ronnquist, A. Hodgson, and A. Lucas, Jack intelligent agents - components for intelligent

agents in java, Technical report, Agent Oriented Software Pty. Ltd., Melbourne, Australia, 1998.

[6] 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.

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

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

[9] J. Castro, M. Kolp, and J. Mylopoulos. ''Towards requirements-driven information systems engineering: The tropos project'', Information Systems, Elsevier: Amsterdam, The Netherlands, 2002.