A Comparative Analysis of Agent Oriented Requirement Engineering Frameworks

Shambhu Bhardwaj
Teerthanker Mahaveer Institute of Management & Technology,
Teerthanker Mahaveer University,
Moradabad, India

Achal Kumar Goyal, Ph.D
Guru Kul Kangri Vishwavidhalaya
Haridwar, India

ABSTRACT

The success of the software system is measured by the degree to which it meets the purpose for which it was intended. Requirement Engineering is the process of discovering that purpose, by identifying stakeholders and their needs and documenting these in a form that is amendable to analysis, communication and implementation. Agent –oriented concepts are becoming very popular in software engineering as modelling frameworks for requirement engineering. This paper introduces the current Agent Oriented Requirement Engineering (AORE) Methodologies. It discusses what approaches have been followed; the suitability of these approaches for agent modelling; compares these approaches in a tabular form and some conclusions drawn from review.

Keywords

Requirement Engineering, Agent Orientation, Software Agent, Modelling Frameworks.

1. INTRODUCTION

The primary measure of success of a software system is the degree to which it meets the purpose for which it was intended. Requirement Engineering (RE) is the process of discovering that purpose. RE is a branch of Software engineering whose ultimate goal is to deliver some system behaviour to its stakeholders. It is receiving increasing attention due to abstract and invisible nature of software and the vast range and variety of problems that admit to software solution. Many approaches to RE are suggested in the literature.

Agent oriented techniques [24] can make a substantial contribution to the implementation of information system by providing additional functionality and better user interface. In this paper we focus on the use of agent oriented framework in the early stages of the design of a system mainly in requirement engineering. Agent concepts have been introduced in RE primarily as modelling constructs to characterize active elements in the environment, usually including the target system. These active elements may be human or machine and may contain hardware or software. In RE, agent orientation brings several important benefits.

First, in AORE the agent’s mental states (beliefs, goals, commitments, etc.) allows modelling at a higher level of abstraction. By assigning such mental states to agents, we may be able to explain or predict their behaviour even when we have little information about their internal control structure. Mental states are also very useful in understanding how the behaviour of agents changes in response to changes in their environment or organization. Second, in modelling the organization or environment in which a system operates, representing communication as various types of ``speech acts'' being performed by agents abstracts over the form and mechanism of messages. Third, Requirements engineering tools can draw on implementation techniques for agent-oriented frameworks to provide more powerful and effective modelling and analysis techniques. Fourth, AORE is motivated by the need for the open architectures that continuously change and evolve to accommodate new components and new requirements. Finally, the move to agent-oriented frameworks is consistent with a long-standing recognition of the need to adequately model the organizations in which information systems operate. In section 2 we review a few selected RE frameworks in which agent play a centric role. Among these Agent Oriented RE frameworks, some are formal and some are informal. Each one is discussed briefly. Section 3 compares the different AORE methodologies in a tabular form. Finally, section 4 concludes the paper.

2. AGENT ORIENTED REQUIREMENT ENGINEERING FRAMEWORKS

In this section, we will discuss briefly some of the agent oriented requirement engineering frameworks.

2.1 i* Modelling Framework

i* modelling framework [5] was given by Eric C. Yu . It was developed for modelling and analysing organization to help support business process re-engineering and requirement engineering. It consist of two main component-i) the strategic dependency model (SD Model) and ii) the static rational model (SR Model). The SD model is use to describe the relationship among the actors. The SR model is used to describe the stakeholder’s interest and concern and how they might be address by various configurations of system and environment. i.e it describe the alternative methods for completing the goal and task for the actors. It helps in understanding the existing processes and in generating the alternatives.
The SD is a diagram based model. It has some nodes and links. Every node represents an actor (here an actor can be an agent, a role or a position) and link between the actor’s represent the dependency. i.e how one actor depends on another for completing a task or a goal. The depending actor is called depender and actor who is depended upon is called dependee. The relation between depender and dependee is called the dependum.

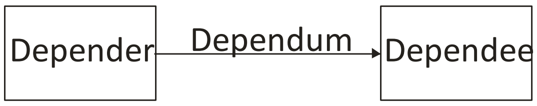

Fig 1: Relation between depender and dependee

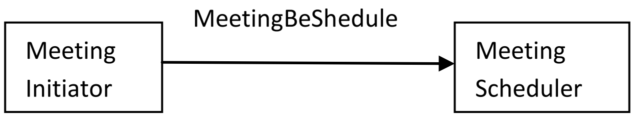

Fig 2: For example in a meeting scheduling process

SD model suggest four types of dependencies: goal dependency, task dependency, resource dependency and soft goal dependency.

SD model focus on the intentional dependency among the actors, instead of the flow of entities among activities [6]. This allows analysis of opportunity and vulnerability. This model does not support the process of suggesting and evaluating the alternative solutions. The SR model addresses this issue. The SR model provides a more detailed level of modeling. It looks inside the actors to model the internal intentional relationship. The intentional elements are goal, task, resource and soft goals. These intentional elements appear in SR model not only as external dependency but also as internal elements. Two types of links are proposed in SR models. These are: - Means-end relationship and task-decomposition. Mean-end link specifies how a goal may be achieved. Task decomposition link specifies how a task can be decomposed into subtask.

2.2 ConGolog Modeling Framework

The ConGolog modeling framework [24] is based on the language ConGolog. The ConGolog language was originated developed as a high level language for programming robots and software agents. ConGolog's semantics is based on situation calculus, a language of predicate logic for representing and reasoning about action. ConGolog model has two components: the first component is a specification of the domain dynamics i.e., how to model the state, what is the initial state of the domain, what actions can be performed, when the actions can be performed and what their effects are? The model can include a specification of a agent's mental states i.e what knowledge and goal they have as well as of the dynamics of these mental states i.e how knowledge and goals are affected by communication actions (eg. Inform, request, cancel request etc.) and perception actions. This component is specified in a declarative way either in high level Golog domain language (GDL) or directly in the situation calculus.

The second component of ConGolog model specifies the behaviour of the agents. This component is specified procedurally. For this there is ConGolog process description language which provides a rich set of constructs for specifying multiagent processes including concurrency, priorities, interrupts and non determinism.

In ConGolog and the situation calculus [5], a domain dynamics is modeled in terms of: agents, primitive actions, situation and fluent entities.

The dynamics of a domain are specified using three kinds of axioms: action precondition axioms, successor sate axioms and initial state axioms.

2.3 An approach to the combined use of i* and ConGolog

i* is an informal diagram based language for early phase requirement engineering and ConGolog is a logic based approach for specifying processes that involves multiple agents. The two approaches are complement to each other. An approach for the combined use of these two methodologies [5] was suggested in requirement engineering. The major steps of this combined methodology are as follows.

3.1.1 *Building the i* SD (Strategic Dependency) model*
3.1.2 *Building the i* SR (Strategic Rationale) model:*
3.1.3 *Building the Annotated Strategic Rationale (ASR) model*
3.1.4 *Developing the initial ConGolog model conforming to mapping rules*
3.1.5 *Validating the ConGolog model by simulation and verification*

Iterate steps 3.1.1 to 3.1.5- Refining the i* and ConGolog models until objectives are met.

Whenever the i* model or ConGolog model has to be modified based on the results of the validation step, the modeler refines the corresponding part of the other model. This continues until the client's objectives are achieved.

2.4 The Requirement Engineering Framework

The REF framework is designed to deal with the WHY, What and HOW of the organizational context [26, 27]. The framework tackles the modeling effort by breaking the activity down into more manageable components and by adopting a combination of different approaches on the basis of a common conceptual notation. Agents are used to model the organization. Goals are used to model the agents's relationship and eventually to link organizational needs to system requirements. Two types of goals are there- Hard goal and Soft goal. A goal is called hard when its achievement criteria are strictly defined ("get marks above 60 %"). In case of a soft goal, the criteria depends on the originator to decide when the goal is considered to have been achieved ("get good marks").In comparison to hard goals, soft goals can be highly subjective and strictly related to a particular context. They enable the analysts to highlight quality issues (e.g. the concept of "good marks") from the outset, making explicit the semantics assigned to them by the stakeholders.

The proposed framework supports three inter-related modelling efforts-

2.4.1 *During Organization Modelling:* the organizational context is analysed and the agents and their goals identified. Any agent may generate its own goals, may operate to achieve goals on the behalf of some other agents, may decide to collaborate with other agents for a specific goal, and might clash on some other ones. The found goals will then be refined, through interaction with the involved agents (i.e. the stakeholders), by hard goal and soft goal modelling.
2.4.2 *The Hard Goal Modelling:* determines how the agent can achieve a hard goal placed upon it, by decomposing them into more elementary subordinate hard goals and tasks. The level of refinement of a hard goal into subordinate hard goals and tasks depends on the level of capability and autonomy of the agent.

2.4.3 The Soft Goal Modelling: aims at producing the operational definitions of the soft goals, sufficient to capture and make explicit the semantics that are usually assigned implicitly by the involved agents and to highlight the system quality issues from the start. A soft goal is refined in terms of subordinate soft goals, hard goals, tasks and constraints. Resulting soft goal, in turn, will have to be progressively refined until a set of hard goals, tasks and constraints is obtained Constraints are associated with hard goals and tasks to specify the corresponding quality attributes.

Three kinds of information flows [26] for dependencies among different agents are suggested in REF-Development flows, Verification flow, and Elicitation & Validation flow. The adopted graphical notation is widely inspired by i* framework and business analysis and re-engineering and thus open to be integrated in or extended by the Tropos methodology.

2.5 Tropos Methodology

The Tropos methodology [7, 10] is intended to support all analysis and design activities in the software development process, from application domain analysis down to the system implementation. In particular, Tropos rests on the idea of building a model of the system-to-be and its environment that is incrementally refined and extended.

Five main development phases are suggested in the Tropos methodology: Early Requirements, Late Requirements, Architectural Design, Detailed Design and Implementation.

The ultimate objective of requirement analysis in Tropos is to provide a set of functional and non-functional requirements for the system-to-be. Requirements analysis in Tropos is split in two main phases: Early Requirements and Late Requirements analysis. Both share the same conceptual and methodological approach. Thus most of the ideas introduced for early requirements analysis are used for late requirements as well. More precisely, during the first phase, the requirements engineer identifies the domain stakeholders and models them as social actors, who depend on one another for goals to be achieved, plans to be performed, and resources to be furnished. By clearly defining these dependencies, it is then possible to state the why, beside what and how, of the system functionalities and, as a last result, to verify how the final implementation matches initial needs.

In the Late Requirements analysis, the conceptual model is extended including a new actor, which represents the system, and a number of dependencies with other actors of the environment. These dependencies define all the functional and non-functional requirements of the system-to-be.

The Architectural Design and the Detailed Design phases focus on the system specification, according to the requirements resulting from the above phases. Architectural Design defines the system's global architecture in terms of sub-systems, interconnected through data and control flows.

The Detailed Design phase aims at specifying 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.

The Implementation activity follows step by step, in a natural way, the detailed design specification on the basis of the established mapping between the implementation platform constructs and the detailed design notions.

2.6 Albert II

Albert II [1, 3] is a formal requirement specification language based on a real time temporal logic. The name is an acronym for Agent-oriented Language for Building and Eliciting Real time requirements. The main purpose of the framework is to model distributed heterogonous real time cooperative system. The development of the language was started in 1992. Throughout its development, the language was tested on specification of non trivial systems like computer integrated system, process control and telecommunication system.

The language [19] is based on the concepts that have been proven useful for capturing functional requirement in real time distribute systems. Its framework is based upon Albert-Core. Its most important characteristic is its naturalness. Naturalness stands for the language ability to map the informal statements provided by the customer onto formal statement expressed in the language.

The Albert II supports the modelling of the functional requirements in terms of collection of agents interacting with each other in order to provide organizational services. Each agent is characterized by actions that may change its state of knowledge of the external world. Actions are performed by the agents to discharge contractual obligations expressed in terms of internal and cooperative constraints.

A specification in Albert II is made up of a graphical part where the vocabulary is declared and a textual part where the logical formulae constraining the admissible behaviour are stated.

Agents in Albert II [9] are not intentional and do not have goals. Albert focuses on specification and is not concerned with the examination of alternatives for meeting goals.

3. COMPARISON OF DIFFERENT AGENT ORIENTED REQUIREMENT ENGINEERING FRAMEWORKS

We have compared the different AORE frameworks based upon some parameters and have obtained the analysis results that are summarized in Table 1. Some frameworks are formal and some are informal. Tropos is the only method that aims at developing new software and thus, its lifecycle coverage goes from early requirements to implementation. i* focuses on early phase while Albert II focuses on late phase requirements. Agents in Albert II are not intentional and do not have goals.

Comparison among AORE Frameworks

	i* Modelling Framework	**ConGolog Modelling Framework**	**An approach to combined use of i* and ConGolog framework**	**The Requirement engineering Framework (REF)**	**Tropos Methodology**	**Albert II**
Informal/ formal Framework	Informal	formal	formal	Informal	Informal	formal
Complete software development methodology	no	no	no	no	yes	no
Agents have goals	yes	yes	yes	yes	yes	no
Agent's participation	design choice and autonomy	constrained behaviour	design choice and autonomy	design choice and autonomy	design choice and autonomy	constrained behaviour
Support for early and late phase requirements	only early phase requirements	no	only early phase requirements	no	yes	only late phase requirements
Agent's goals are same	no	yes	yes	no	no	yes
Example	meeting scheduler system	meeting scheduler system	meeting scheduler system	electronic record management system	e-cultural system	generalized railroad crossing problem
Resources produced	complete SD and SR models	ConGolog Model	implementation using ConGolog tool	a set of hard goals, tasks and constraints	system implementation	system requirement document
Intermediate artifacts	SD and SR model before considering computer based system	ConGolog procedures	SD Model, SR Model and ASR Model	organizational modelling, hard goal modelling and soft goal modelling	capability diagram, plan diagram, agent interaction diagram	graphical declaration and textual specification of constraints
Implementation tool	–	ConGolog tool	–	–	JACK tool	–

4. CONCLUSION

In this paper, we have reviewed a number of Agent Oriented RE frameworks. We presented the different frameworks of RE and the advantages of AORE over these frameworks. Also we discussed the different frameworks under the umbrella of AORE and compare them in a tabular form.

We have presented six different AORE frameworks and analysed them in order to inform their use. The comparison focuses on the process, the resources involved and RE issues addressed by each method.

As a result, and this is the main contribution of our work, this comparison show under which circumstances one method may be more valuable than the others and therefore their selection may now depend upon the objective criteria rather than on a subjective belief. Further work includes a deeper analysis of these frameworks by evaluating the comparison from different points of view (e.g., the authors, students, industry practitioners and developers). Also a quantitative approach should be adopted to allow a numerical evaluation. A single case study can be used for the comparison of these frameworks.

We should note that the move to agent-oriented frameworks is consistent with a long-standing recognition of the need to adequately model the organizations in which information systems operate.

5. REFERENCES

[1] Du Bois, P., The Albert II Language - On the Design and the Use of a Formal Specification Language for Requirements Analysis, Ph.D. thesis, Dept. of Computer Science, University of Namur, Namur, Belgium, 1995.

[2] Jennings, N.R. and Wooldridge, M. (Eds.), Agent Technology: Foundations, Applications, and Markets, Springer-Verlag, Berlin, 1998.

[3] Alexei Lapouchnian , "Modeling Mental States in Requirements Engineering – An Agent-Oriented Framework Based on i* and CASL", A thesis submitted to the Faculty of Graduate Studies in partial fulfillment of the requirements for the degree of Master of Science York University Toronto, Canada July, 2004

[4] Shapiro, S., Lespérance, Y., and Levesque, H.J., Specifying Communicative Multi-Agent Systems with ConGolog, in Agents and Multi-Agent Systems - Formalisms, Methodologies, and Applications, W. Wobcke, M. Pagnucco, and C. Zhang, eds., 1-14, LNAI, Springer-Verlag, Berlin, 1998.

[5] Wang, X., Agent-Oriented Requirements Engineering Using the ConGolog and i* Frameworks,M.Sc. thesis, Dept. of Computer Science, York University, Toronto, ON, Canada, 2001, to appear.

[6] Yu, E.S.K., Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering, in Proc. of the 3rd IEEE Int. Symp. on Requirements Engineering (RE'97), 226-235, Washington, DC, 1997.

[7] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. TROPOS: An agent oriented software development methodology. Autonomous Agents and Multi-Agent Systems, 2003. in Press

[8] E. Yu. Modeling Strategic Relationships for Process Reengineering. PhD thesis, University of Toronto, Department of Computer Science, University of Toronto, 1995.

[9] E. Yu. Why agent-oriented requirements engineering. In Proceedings of 3rd Workshop on Requirements Engineering For Software Quality, Barcelona, Catalonia, June 1997.

[10] J.Castro, M.Kolp, J.Mylopoulos. Towards requirements driven information system engineering: The Tropos Project.

[11] D'Inverno M., Luck, M. "Development and Application of an Agent Based Framework" Proceedings of the First IEEE International Conference on Formal Engineering Methods, Hiroshima, Japan, 1997.

[12] Rumbaugh, J., Jacobson I., Booch, G. The Unified Modelling Language Reference Manual. Rational Software Corporation, Addison Wesley, UK, 1999.

[13] Colette Rolland, Carine Souveyet, and Camille Ben Achour, Guiding Goal Modeling Using Scenarios, IEEE Transactions On Software Engineering, Vol. 24, No. 12, December 1998

[14] J. A. Bubenko, Extending the Scope of Information Modeling, Proc. 4th Int. Workshop on the Deductive Approach to Information Systems and Databases, Lloret-Costa Brava, Catalonia, Sept. 20-22, 1993, pp. 73-98.

[15] M. Jackson, System Development, Prentice-Hall, 1983

[16] Carlos A. Iglesias , M. Garijo, " A Survey of Agent Oriented Methodologies"

[17] Bubenko JA. Information modeling in the context of system development. In S.H. Lavington, editor, Information Processing 80, pages 395-411. North-Holland, 1980

[18] S. Nwana "Software agents: An Overview" (1996)

[19] Phillippe Du Bois, Eric Dubois, Jean- Marc Zeippen, " On the Use of a Formal Requirement Engineering Language: The Generalized Railroad Crossing Problem" , Springer-Verlag London Limited -1997

[20] Awais Rashid, Peter Sawyer, Ana Moreira, João Araújo, Early Aspects: a Model for Aspect-Oriented Requirements Engineering

[21] Ecole Polytechnique Federale de Lausanne, Goal Driven Requirements Engineering Overview

[22] Smith, R. (1996a), "Software Agent Technology", Proceedings of The First International Conference on the Practical Applications of Intelligent Agents and Multi-Agent Technology, London, UK, 557-571

[23] Bashar N. and S. Easterbrook, "Requirement Engineering: A Roadmap"

[24] Y. Lesperance , Steven Shapario, "On Agent Oriented Requirement Engineering"

[25] E. Yu and J. Mylopoulos, Understanding Why in Requirements Engineering – with an Example,Workshop on System Requirements: Analysis, Management, and Exploitation, Schloß Dagstuhl, Saarland, Germany, October 4–7, 1994.

[26] Paolo Donzeli, " Agents, goals and Quality in a Structured Requirement Engineering Framework-a case study"

[27] Paolo Bresciani and Paolo Donzeli, " REF: a Practical Agent Based Requirement Engineering Framework"

[28] Amund Tveit "A survey of Agent-Oriented Software Engineering" (2001)

[29] Lespérance, Y. and Shapiro, S., On Agent-Oriented Requirements Engineering, position paper, International Workshop on Agent-Oriented Information Systems (AOIS'99), Heidelberg, Germany, June 1999.

[30] Nancy R. Mead, Eric D. Hough, Theodore R. Stehney II, Security Quality Requirements, Engineering (SQUARE) Methodology, November 2005

[31] S. Ratchev, E. Urwin, D. Muller, K.S.Pawar, I. Moulek, Knowledge based requirement engineering for one-of-a-kind complex systems, Jan 2002