Integrative Early Requirements Analysis for Agent-Based Systems

Longbing Cao[1], Chengqi Zhang[1], Dan Luo[2], Wanli Chen[1], Neda Zamani[3]

[1]Faculty of Information Technology, University of Technology, Sydney, Australia
[2]Science and Technology Exchange Center, Ministry of Science and Technology, China
[3]Faculty of Informatics, University of Wollongong, Wollongong, Australia
[1]{lbcao, chengqi, wanli.chen}@it.uts.edu.au, [2]dan.luo@mail.ia.ac.cn, [3]nz47@uow.edu.au

Abstract. Early requirements analysis (ERA) is quite significant for building agent-based systems. Goal-oriented requirements analysis is promising for the agent-oriented early requirements analysis. In general, either visual modeling or formal specifications is used for the ERA. This way cannot capture requirements precisely and completely. In this paper, we present an integrative modeling framework for agent-oriented early requirements analysis; this framework implements goal-oriented requirement analysis. The integrative modeling combines visual modeling and formal modeling together. Extended *i** framework is used for building visual models; formal specifications complement the visual modeling to define and refine requirements. Both visual and formal models are outlined through a practical agent-based system F-TRADE[1]. The integrative modelling seems to model early requirements comprehensively and concretely, and benefit refinement and conflict management in building agent systems.

Keywords: agent-oriented requirements analysis, goal-oriented requirements analysis, integrative modeling, visual modeling, formal modeling

1 Introduction

Early requirements analysis (ERA) plays significant role in the process of building agent-based systems. One promising approach is to utilize goal-oriented requirements analysis (RA) (Dardenne et al. 1993, Lamsweerde 2001) to do this work. An agent (Wooldridge 2001) is a goal-driven entity with autonomy and self-control of capabilities. Goal-oriented analysis is compatible with the motivation of agent-oriented methodology, and is becoming involved in agent-oriented requirement engineering.

Visual modeling with diagrammatic specifications is widely used in both goal-oriented and agent-oriented requirements analysis, for instance, AUML[2] and the *i** framework (Yu 1993). Nevertheless, this may lead to subjective models for lacking a formal definition of their semantics, which can hardly be refined in a straightforward way into the phase of system design. Formal specifications can complement some of weaknesses of visual modeling with precise representation. However, an integrative model with both visual and formal representations presents complete and precise information in early requirement engineering.

In this paper, we present goal-oriented integrative modeling for agent-oriented early requirements analysis. The integrative modeling consists of both visual modeling and formal modeling. A visual model and formal specifications are interdependent and complement each other. Organizational Dependency model and Organizational Rationale model are deployed for visual modeling by extending the *i** framework. First-order linear-time temporal logic is used for formal analysis and refinement. In integrative modeling, both functional and nonfunctional requirements can be analyzed.

An agent service-based open infrastructure called F-TRADE is taken as the case study for the proposed modeling approach. F-TRADE has been developed for trading and mining supports in real capital markets. The success of requirement engineering and system implementation of F-TRADE shows that the integrative modeling can make agent-oriented early requirement engineering effective and efficient.

The sequel of this paper is organized as follows. Section 2 briefly introduces related work. In Section 3, a framework of integrative modeling is given for agent-oriented early RA. Section 5 and 6 discuss Visual modeling and formal specifications, respectively in terms of the case study F-TRADE in Section 4. Section 7 sketches system implementation of the F-TRADE using agent service technology. Finally, Section 8 summarizes the contributions of the paper, and presents

[1] F-TRADE (http://datamining.it.uts.edu.au:8080/tsap) gets real data and industrial requirements in capital markets from Capital Markets Cooperated Research Center, Australia (www.cmcrc.com).

[2] www.auml.org

future work.

2 Related Work

Goal-oriented analysis focuses on the description and evaluation of alternatives and their relationships to the organizational objectives behind a software project. Capturing these interdependencies between organizational objectives and software requirements can facilitate the tracing of the origins of requirements, and can help make the requirements process more thorough, complete and consistent.

Many techniques have been proposed for goal-oriented requirement engineering. Some goal-oriented approaches link agents and goals together. In KAOS (Dardenne et al. 1993), *responsibility* links are introduced to relate the goal and agent sub-models. A goal may be assigned to alternative agents through responsibility links; this allows alternative boundaries to be explored between the software-to-be and its environment. In the *i** framework, agent dependency links are defined to model situations where an agent depends on another for a goal to be achieved, a task to be performed, or a resource to become available.

Most of goal-oriented modeling techniques use graphical notations; this can be called as Diagrammatic Modeling or Visual Modeling. Diagrammatic modeling makes the scenarios more understandable. The *i** framework uses graphical notations to analyze requirements. *i** models will be built for early requirements analysis, and Strategic Dependency model as well as Strategic Rationale Model are deployed for late requirements analysis. However, they enclose implicit, incomplete and somehow imprecise information which undermines the understanding of the problem and blocks later design and implementation. To this end, some proper formal specifications may complement them.

A new trend is to integrate visual modeling and formal specifications. (Koning et al. 2002) proposed the ATOS approach; it introduces a textual notation of AUML that can be translated to an extended finite state machine to be processed by a model checker. It is reported that it has been used to perform formal verification of AUML sequence diagram specification of interaction protocols of multi-agent systems. (Perini et al. 2003) discussed the possibility of inter-mixing formal and informal specifications in order to guide and support the conceptual modeling; it described a modeling language for lightweight usage of formal verification techniques when performing conceptual modeling in agent-oriented methodology.

Most of the proposed methodologies in goal-oriented RA adopt visual modeling as a core process. Visual modeling absorbs and inherits some best practices like structured object-oriented software engineering. For instance, "use case" is used for discussing requirements with actors; *i** framework and Tropos methodology (Lamsweerde 2000) provide graphical notations for describing actors, goals, dependencies, rationale, and so forth. The usage of visual modeling languages in requirements analysis offers advantages such as that of providing an effective and concrete interaction for different stakeholders of the process.

However, visual modeling languages lack a formal definition of their semantics. This could lead to subjective models, which can hardly be refined in a straightforward way into a system design. Another weakness is when the refinement should be stopped in building a conceptual model.

Formal specifications overcome some of the weaknesses of visual modeling. It presents mechanisms for defining models with precise semantics. The disadvantages of formal specifications include that (i) it is hard for understanding and utilizing in modeling without strong skills in the specifications, and (ii) it is not effective in visualizing interactions between stakeholders. However, these can be handled by visual modeling. So, the integration of visual modeling and formal modeling can complement each other, and make the RA as understandable and precise as possible.

3 Integrative Modeling Framework

Integrative Modeling is a methodology which tries to integrate formal specifications with informal diagrammatic notations; it also integrates functional requirements and nonfunctional requirements in the process of ERA.

3.1 Integrating requirements of functional and nonfunctional

Functional requirement analysis

The functional requirements analysis concerns with the understanding of goals of a system by studying an organizational setting. The output of this analysis is an organizational model which includes relevant actors, their respective goals and their inter-dependencies.

To this end, we try to adopt concepts offered by *i** framework, such as actor (actors can be agents, positions or roles), as well as social dependencies among actors, including goal, soft goal, task and

resource dependencies. Necessary extension will be discussed in later sections.

Nonfunctional requirement analysis

Nonfunctional goals (or quality attributes, qualities, or more colloquially "-ilities") refer to quality of service, development objectives or architectural constraints (Lamsweerde 2003). Most common nonfunctional requirements (Mylopoulos et al 1999) are global qualities of a software system, such as flexibility, maintainability, usability, and so forth. The characteristics of such requirements include (1) they are usually stated only informally, (2) they are often controversial (for example, management wants a secure system but staff desires user-friendliness), (3) they are difficult to enforce during design and implementation, and (4) they are difficult to validate.

Integrative requirements analysis

In a summary, functional requirements focus more on functional goals; while nonfunctional requirements cares about nonfunctional goals. Nevertheless, functional goals and nonfunctional goals are closely related or even interleaved in a system. It is to some degree hard to separate them from each other in a real system even though we often discuss them in different situations and for different objectives. For instance, the system-to-be is described along with relevant functions and qualities within its operational environment.

On the other hand, even though we can build individual models for functional and nonfunctional requirements respectively, the representation of these two types of models can be in same or similar diagrammatic descriptions. In goal-oriented requirements analysis, both nonfunctional and functional requirements analyses can be described by modeling techniques like Strategic Dependency Model and Strategic Rationale Model in *i** modeling.

An *Integrative Requirements Analysis*, which must and can enclose both functional and nonfunctional requirements, is more practical and effective in both modeling and understanding of the problem. It will further benefit the later system design and implementation.

Some weaknesses of integrative modeling include too many goals will be involved in one model, and the diagram could be very complicated and comprehensive. However, as we use decomposition technique to understand a complex system by dividing it into multiple subsystems, functions and qualities can be thought about separately first and then united into an entity. In order to mark functional and nonfunctional requirements in the integrated diagrams, different diagrammatic notations will be used to label them respectively. This can help to trace and understand the two-type goals according to different motivations.

3.2 Integrative modeling of visual models and formal specifications

Visual modeling

Most of the proposed methodologies in goal-oriented requirements analysis adopt visual modeling as a core process. Visual modeling absorbs and inherits some best practices like structured object-oriented software engineering (Kruchten 2000). For instance, use case (Leffingwell et al 2003) is used for discussing requirements with actors; *i** framework and Tropos methodology provide graphical notations for describing actors, goals, dependencies, rationale and so forth. The usage of visual modeling languages in system analysis and design offers advantages such as that of providing an effective and concrete interaction for different stakeholders of the process.

Nevertheless, visual modeling languages lack a formal definition of their semantics. This could lead to subjective models, which can hardly be refined in a straightforward way into a system design. Some other weaknesses include when the refinement should be stopped in building a conceptual model, and how to check the conflict and inconsistencies.

Formal specifications

Formal specifications, however, complement some of weaknesses of visual modeling. It presents mechanisms for defining models with a precise semantics. The disadvantages of formal specifications include that it is hard for understanding and utilizing in modeling without strong skills in the specifications, and it is not good at visualizing interactions between stakeholders.

Integrative Modeling

An *Integrated Modeling* is a kind of modeling methodology which tries to integrate formal specifications and informal diagrammatic notations systematically. Visual modeling equipped with formal languages can interpret the problem and handle requirement engineering in a concrete and visualized way, which helps to discuss with stakeholders; it can also support deductive reasoning which assists with precise and explicit understanding of the problem and modeling procedures.

In order to build integrative models, we extend *i** framework, and take it as visual modeling technique

for the functional and nonfunctional (Mylopoulos 1999) requirement analyses in our work (Cao 2005). At the same time, the real-time linear temporal logic (Manna et al. 1992), also presented in Formal Tropos (Lamsweerde 2000), will be used to deploy a formal specification for describing organizational elements and relationships formally and explicitly. Figure 1 depicts a framework for the integrative modeling technique. The following sections will discuss the integrative modeling by analyzing a real case.

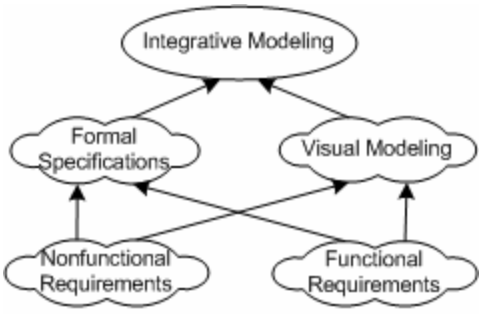

Figure 1. Integrative modeling

4 A Case Study

F-TRADE (Financial Trading Rules Automated Development and Evaluation) (Cao 2005, Cao et al. 2004) is a web-based trading and mining support infrastructure. It supports online plug-in of trading strategies or data mining algorithms and fully observes privacy. Providers can iteratively evaluate these algorithms with online connectivity to real stock data from global markets, and even find some optimum strategy before going to markets. Public traders and investors can benefit from available strategies on F-TRADE by subscription. Actually, F-TRADE also supports plug in of data sources, and system functional modules. Moreover, optimal strategies or optimal combination of input parameters for a specific trading strategy can be recommended on F-TRADE.

4.1Visual OR model

For modeling the F-TRADE, we adopt and extend the *i** framework as visual modeling (Cao 2005). The original Strategic Dependency model is expanded to be Organizational Dependency (OD) model; the Strategic Rationale model becomes Organizational Rationale (OR) model for describing agent systems in organizational metaphor. For space limitation, we only discuss about OR model in this paper.

The OR model introduces instantiated organizational actors, relationships and rules. Goals and tasks in high granularity in an OD model will be decomposed and refined into grain sub-goals and sub-tasks. Correspondingly, more concrete and detailed responsibilities, specific relationships, and causalities will be discovered and exhibited in the model. Task decomposition can further be expressed by Sister Relationships between subtasks (or sub-goals) and Parent-Child Relationships between super-task (or super-goal) and subtask (or sub-goal) (Cao 2005). We introduce how to build the OR model through the case study as follows.

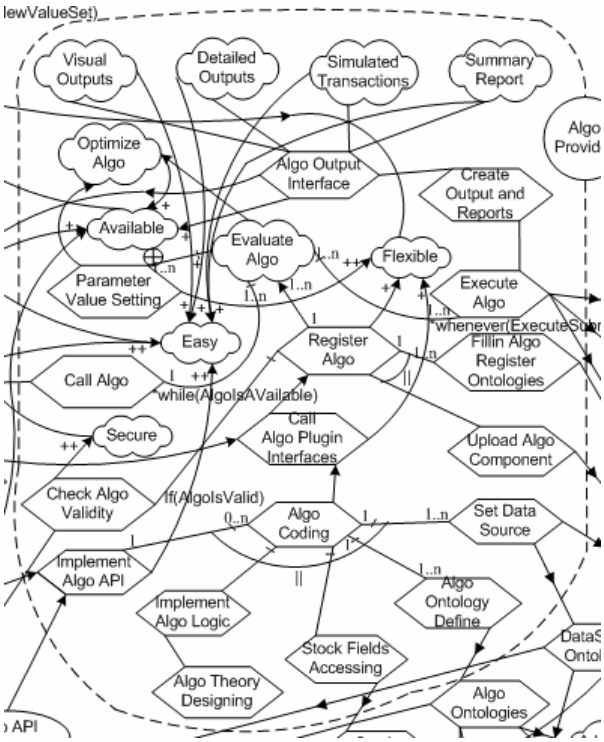

Figure 2. Excerpt of OR model for algorithm *Provider*

Figure 2 shows an excerpt of the OR model for organizational actor *Algorithm Provider* and its environment in F-TRADE. The main work for an algorithm provider is to code her/his algorithm, register it into F-TRADE, and evaluate it as s/he likes. In the center of the figure, one basic task *Algo Coding* is refined into *Implement Algo API*, *Implement Algo Logic*, *Stock Fields Accessing*, *Algo Ontology Define*, and *Set Data Source* concurrently. After programming of an algorithm, the provider can log onto F-TRADE and *Call Algo Plugin Interfaces* to *Register Algo*. *Register Algo* will be further decomposed into *Fillin Algo Register Ontologies* and *Upload Algo Component* in parallel. After *Check Algo Validity* by *Administrator*,

Algo Provider goes to *Evaluate Algo*. This includes subtasks as *Call Algo*, *Parameter Value Setting*, *Execute Algo*. The *Algo Output Interface* will be generated automatically, and shows outputs and reports in *Visual Outputs*, *Detailed Outputs*, *Simulated Transactions*, and *Summary Report*, respectively.

Partially nonfunctional organizational rationale is also shown on the figure. Softgoal contributions are also presented to model sufficient/partial positive (++ and + respectively) or negative (-- or - respectively) support to softgoals *Flexible*, *Available*, *Secure*, *Easy* and *Adaptable*. For instance, to provide sufficient support to the soft-goal *Flexible*, some functional tasks are specified to deal with *registering algorithm*, *calling algorithm plugin interfaces*, and *iteratively setting parameter values*.

The OR model does help us examine the rationale of the system itself more explicitly and precisely. With the cooperative deployment of the OD and the OR models, functional and nonfunctional requirements can be analyzed and refined up to relevantly detailed and complete understanding of the system. However, a more precise elaboration can only be acquired by developing formal specifications.

4.2 Formal analysis of OR model

Formal specifications for agent-oriented requirements analysis are described by formal grammar. Formal grammar can be defined by the first-order linear-time temporal logic. In the following, the main operators from the first-order linear-time temporal logic are presented. Part of the temporal formulae used in this work is also described. More information about formal specifications can be obtained from (Cao 2005).

A formal analysis can be performed to refine an early requirements specification with the above formal grammar. In general, formal analysis will be an iterative process. In formal analysis, the OD model and OR model will be built and refined gradually by mapping actors and intentional elements into corresponding formal entities and by adding non-intentional entities of the domain, if any.

Here we just take the above excerpt of OR model as an example for the formal analysis. The goal *RegisterAlgo* for registering an algorithm by *Algorithm Provider* into F-TRADE will be taken as an example for introducing formal modeling. With the goal- and scenario-based (Sutcliffe 1998, Cao 2005) modeling techniques, we can get a refinement of goal *RegisterAlgo*. The Creation condition for an instance of goal *RegisterAlgo* is that its predecessor goal *CodeAlgo* has already been fulfilled. The invariant shows constraints on the lifetime of class instances. For goal *RegisterAlgo*, the invariant binds a *RegisterAlgo* object with its predecessor object. To fulfill goal *RegisterAlgo*, for each algorithm, there exists only one *AlgorithmComponent* instance that has been coded and will be uploaded at sometime t_2 in the future through calling *CallPluginInterfaces* at future time t_1 ($t_1 \le t_2$); one legal instance of *FillinAlgoRegisterOntologies* is filled at t_2. The *FillinAlgoRegisterOntologies* are allowed for multiple instances by following rules in the class *Ontology* for each *AlgorithmComponent*. The excerpt of the refined formal specification for goal *RegisterAlgo* is shown in Figure 3, where both informal and formal definitions are given.

Goal **RegisterAlgo**
InformalDef *When an algorithm component has been coded and the algorithm isn't available from the system at the moment, this algorithm component can be registered into the system by calling plug-in interfaces, filling in algorithm registration ontologies, and upload the algorithm module.*
FormalDef
Actor **Provider**
Mode achieve
Attribute constant **ca**: **CodeAlgo**
Attribute constant **algo**: **Algorithm**
registered: boolean
Creation condition
● Fulfilled(**ca**) ∧ ¬ Existed(algo)
Invariant **ca**.actor = actor
Fulfillment condition
∀ ac: **AlgorithmComponent (ac.**algo = algo →
$\Diamond_{\le t1}$ ∃ c**pi**: **CallPluginInterfaces (cpi**.actor= actor ∧ Fulfilled(**cpi**) ∧ **pi**.**Called)** ∧
$\Diamond_{\le t2}$(∃**faro: FillinAlgoRegisterOntologies (faro.**depender = actor ∧ Fulfilled(**faro**) ∧ **aro**.**Filled)**
∧∃ **uac: UploadAlgoComponent (uac.**depender = actor ∧ Fulfilled(**uac**) ∧ **ac**.**uploaded)))**

Figure 3. An excerpt of formal specifications for *provider* to register algorithm

As shown in the above case study, formal modeling can benefit responsibility refinement and conflicts management (Lamsweerde et al. 1998), for instance, various kinds of inconsistency resulting from the acquisition, specification, and evolution of goals/requirements from multiple sources, in requirement engineering.

5 Conclusions and Future Work

We have presented the integrative modeling approach for agent-oriented ERA. The main contributions of this paper include:

(1) Introducing the integrative modeling framework by combining visual modeling and formal modeling in terms of goal-based functional and nonfunctional RA.

(2) Outlining the integrative modeling through a practical agent system F-TRADE by extended *i** framework and first-order linear-time temporal logic.

More work is currently being performed on responsibility refinement and model checking. We are also using the integrative modeling strategy on late RA in building agent systems.

Acknowledgments

Thanks are due to CMCRC[3] and AC3[4] Australia who donate a large amount of real data for our experiments.

References

Cao, L. (2005 to appear), "Agent services-oriented analysis and design", PhD thesis. University of Technology Sydney, Ausrtalia.

Cao L., Wang J., Lin L., and Zhang C. (2004), "Agent Services-Based Infrastructure for Online Assessment of Trading Strategies", *proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, IEEE Computer Society Press.

Dardenne, A., van Lamsweerde, A., and Fickas, S. (1993), "Goal-directed requirements acquisition", *Science of Computer Programming 20*, 3–50.

Java Community Process (JCP), *Java agent services specification*, 5 Mar 2002.

Koning J.L., Romero-Hernandez I. (2002), "Generating Machine Processable Representations of Textual Representations of AUML"., *AOSE*.

Kruchten 2000 Philippe Kruchten. *The Rational Unified Process: An Introduction*. Addison-Wesley, 2nd edition.

Lamsweerde A.V. (2000), "Formal Specification: a Roadmap", *In The Future of Software Engineering*, A. Finkelstein (ed.), ACM Press, pp. 147-160.

Lamsweerde, A.V. (2001), "Goal-Oriented Requirements Engineering: A Guided Tour", *Proceedings RE'01, 5th IEEE International Symposium on Requirements Engineering*, Toronto, 249-263.

Lamsweerde 2003 Axel van Lamsweerde. From System Goals to Software Architecture. In Formal Methods for Software Architectures, M. Bernardo & P. Inverardi (eds), LNCS 2804, Springer-Verlag, 2003.

Lamsweerde V., Darimont R. and Letier E. (1998), "Managing Conflicts in Goal-Driven Rquirements Engineering", *IEEE Transactions on Software Engineering, Special Issue on Managing Inconsistency in Software Development*, Vol. 24 No. 11, pp. 908 - 926.

Leffingwell et al 2003 Dean Leffingwell, Don Widrig.Managing Software Requirements: A Use Case Approach, Second Edition, Addison-Wesley Pub Co.

Manna Z. and Pnueli A. (1992), *The Temporal Logic of Reactive and Concurrent Systems*, Springer-Verlag.

Perini A., Pistore M., Roveri M., Susi A. (2003), "Agent-oriented modeling by interleaving formal and informal specification", *AOSE*.

Mylopoulos J., Chung L., and Yu E. (1999), "From Object-Oriented to Goal-Oriented requirements analysis", *COMMUNICATIONS OF THE ACM*, Vol. 42, No. 1, p31-37.

Sutcliffe A. (1998), "Scenario-Based Requirements Analysis", *Requirements Engineering Journal* Vol. 3 No. 1, 48-65.

Wooldridge, M. (2001), *An introduction to multiagent system*, Wiley.

Yu, E.S.K. (1993), "Modelling Organizations for Information Systems Requirements Engineering", *Proc. RE'93 - 1st Intl Symp. on Requirements Engineering, IEEE,* 34-41.

Zambonelli F., Jennings N.R. and Wooldridge M., (2003) "Developing multiagent systems: the Gaia Methodology", *ACM Trans on Software Engineering and Methodology*, 12 (3) 317-370.

[3] www.cmcrc.com

[4] www.ac3.com