Why Agent-Oriented Requirements Engineering

Eric S. K. Yu

Faculty of Information Studies, University of Toronto
Toronto, Ontario, Canada M5S 3G6
www.cs.utoronto.ca/~eric

Abstract. Agent concepts have been used in a number of recent approaches to requirements engineering. In view of the rapid shift towards open, networked, and cooperative computing, we argue for the fuller development of emerging agent-oriented approaches to requirements engineering. Such approaches would address crucial requirements engineering concerns such as functionality, quality, and process, using *agent* as the focal concept.

1 Introduction

Agent concepts have been used in a number of recent approaches to Requirements Engineering (RE). It is generally acknowledged that the main focus of Requirements Engineering should be on the characterization of the intended system in relation to its environment [2, 12, 14]. 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 and/or software.

In this paper, we put forth the position that the concept of agent should be further developed to serve as a focal, guiding concept in RE frameworks, much like *objects* and *goals* have served as guiding concepts. The key benefit of having an RE-level concept of agent, and using it as a guiding concept during RE, is that it will serve to bring issues centring on an agent together, so that they can be identified and addressed.

An agent-oriented approach to RE will be of particular interest for new settings in which there is a high degree of open distributed computing [13, 17], and in which change is ongoing [6]. In agent-oriented RE frameworks, crucial RE concerns such as functionality, quality, and process will be organized around agents so that they will be addressed in a way which is appropriate for open, distributed, and constantly evolving environments.

In section 2, we briefly review a few selected RE frameworks in which agent concepts play a substantial role. In section 3, we outline a number of issues that need to be investigated in order to further develop an agent-oriented approach to RE.

2 Agent Concepts in Requirements Engineering frameworks

We will review briefly the concepts of agent as used in several selected RE frameworks.

(a) Composite Systems Design and KAOS

A composite system is one that is made up of a number of components – or agents – that interact with each other to produce some overall behaviour. An important premise of the CSD approach is that in designing an automated system, one should treat the system and its environment as a (larger) system whose overall properties are the ones we aim to achieve. The specification of the automated system should be derived systematically from the desired behaviour of the overall system. Typically, one starts with global system goals, which are decomposed until they can be assigned to individual agents [9, 10].

Agents have capabilities, which means that their behaviour (sequences of activities) can be constrained so that desired properties are satisfied. During the design process, goals are replaced by 'responsibilities' and assigned to subsets of agents. Ultimately, responsibilities are subdivided so that individual pieces are assigned to individual agents.

KAOS is a framework for goal-directed requirements acquisition [5]. As in CSD, a composite system viewpoint is adopted. Agents may be human, hardware or software. The concept of agent is similar to that in CSD. An agent is a specialized kind of object. Like other objects, it has states and other properties associated with objects.

However, it is also a processor for some actions, and therefore controls state transitions. Unlike other kinds of objects (which include entities, relationships, and events) agents have choice over their behaviour. Each agent has capabilities, which is a list of actions that the agent can perform. The designer starts with goals for the overall composite system, then refines them until they are reduced to constraints that can be assigned to agents.

Agents can have wishes that are goals, and which may conflict with each other. Agents have further attributes to represent load, reliability, cost, and motivation. These are used by the designer in deciding what constraints can be assigned to what agents.

(b) Albert II

The Albert language supports the modelling of functional requirements in terms of a collection of agents interacting in order to provide services necessary for an organization. Each agent is characterized by actions that change or maintain its own state of knowledge about the external world and/or states of other agents. Such actions are performed by agents in order to discharge contractual obligations expressed in terms of internal and cooperation constraints [7, 8].

Functional requirements in Albert are expressed in terms of a set of formal statements. These statements are grouped around agents in order to define the set of admissible behaviour the agents may experience. Thus the notion of agent is seen as a way of organizing the specification so that behaviour pertaining to each (class of) agent is collected together for readability. Similarly, distinguishing constraints that are local to an agent from those that relate to other agents facilitates understanding of the agent's behaviour.

Agents in Albert are not intentional and do not have goals. The language focuses on specification, and is not concerned with the examination of alternatives for meeting goals.

(c) The F3 framework

The F3 framework (From Fuzzy to Formal) [3] is a framework which aims to cover a wide range of requirements engineering activities. An important part of the framework is the "Enterprise Model". This model has five submodels: Objective Model, Concept Model, Activities and Usage Model, the Actors Model, and Information Systems

Requirements Model. These models are essentially ERA models with entities, relationships, and attributes suitable for each of these subareas. There are also links across these submodels of the overall model.

The approach is objectives-driven, with an emphasis on understanding the business objectives behind system requirements. The Objectives model has nodes for Goals, Problems, Opportunities, Causes, Rules, and Development-Actions, which are related via Motivates and Influences links. The latter can be of type Positive, Negative, or Unknown. The contents of these nodes are informal natural language statements.

The Actors submodel have nodes that are Organizational-Units or Roles, which are related via links such as Reports-To, Belongs-To, Supplies-Software-To, and Develops-System-For. Actors can be humans, computing devices or other non-human resources. The Actors model is connected to the Objectives Model via Motivates links, to the Activities and Usage Model via Performed-By type of links, and to the Information Systems Requirements Model via Concerns type of links. The overall Enterprise Model is linked to the more formal Information Systems Model via Implemented-By and Supported-By types of relationships.

(d) The i^* modelling framework

$\boldsymbol{i^*}$ is a framework for modelling and redesigning intentional relationships among strategic actors [18, 20]. It was developed as a framework to support the early phase of requirements engineering. The strategic actor is the central concept. Actors have intentional properties such as goals, beliefs, abilities, and commitments. They are strategic in that they are concerned about opportunities and vulnerabilities.

The focus of the framework is in analyzing strategic implications from the viewpoint of each actor, and to support the exploration and identification of alternative operational processes (solutions) which will better meet the strategic interests of the actors. To support this level of modelling and reasoning, a process is modelled in terms of intentional relationships among actors, rather than input/output relationships. Actors depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished. This is called the Strategic Dependency model.

The Strategic Rationale model provides a way for systematically asking why a process is structured the way it currently is, thus revealing the goals behind them. These may be revised. New solutions are then sought for addressing the goals. The solutions are thus redesigned operational processes which are typically rearrangements of the strategic dependency relationships among actors, and may also involve omission of some actors, introduction of new ones, or merging or splitting of responsibilities.

The framework is not concerned with the detailed specification of processes and activities, leaving these to a separate part of requirements engineering, called the "late-phase".

3 Research Issues for Agent-Oriented RE

Comparing and contrasting the above RE frameworks raises a number of issues. These include the ontological status of "agents" in RE (as opposed to in Systems Design or Implementation), relationship of the agent concept to other important guiding concepts in RE, particularly: goals, quality, process, and object, and how agent concepts can best complement these concepts. Two other important issues are scalability, and the ability to deal with change. These issues are discussed in this section.

3.1 A distinctive notion of agent for RE?

In order to clarify the notion (or notions) of agent that might be useful in RE, one of the central questions is:

> Is there (or should there be) a notion of agent that is distinctive for RE?

For an RE framework to be truly agent-oriented, the identity and existence of an agent (as a construct in the RE level) needs to be determined within the RE level, based on RE criteria. This will allow the process of identifying and delineating agents to produce insights that will inform the RE process. This is in contrast to doing RE for an agent-oriented *system*, in which the identities of the agents are determined according to Implementation level criteria. If agent identity and existence are pre-determined, the RE process may not be benefitting much from having an agent construct.

The desire for ontological independence, however, poses problems when we want to deal with brown-field situations, where there are existing agents (whose identity are therefore determined by concrete existence, i.e., at the Implementation level). Customarily, distribution is a rather "physical" concept. The term "multi-agent systems" typically refers to systems which are implemented with multiple physical components. However, if distribution is an issue that is to be explored at the RE level, then it needs to be a higher-level "logical" concept. For example, task forces, committees, project teams, strategic alliances, are all examples of "logical" agent-like entities in the context of human organizations.

One approach is to offer several variants of agent notions to represent different degrees of concreteness. In $\boldsymbol{i}^*$, the term *agent* is used to refer to the concrete, implementable variety, and therefore whose identity is determined by physical and implementability criteria. *Roles* are abstract. During RE, roles serve as holders of intentionality, so that they can be reasoned about (and reconfigured, i.e., "redesigned", within the context of RE) before they are eventually assigned to agents. A third variant, *position*, being a bundling of roles which are typically assigned to an agent as a unit, serves as an intermediate mapping between agents and roles. Thus one can reason about the design of a position, independently of its eventual occupation by some agent. In $\boldsymbol{i}^*$, the term *actor* is used to refer to the generic concept. Agent, role, and position are specializations of the actor concept. In the rest of this paper however, we will continue to use the term agent in the more general sense, with the understanding that further distinctions such as roles and positions may be helpful.

A distinctive RE agent can serve as a powerful abstraction mechanism – a locus for identifying and addressing issues pertaining to that agent, and how these issues will propagate to other agents. Since issues addressed at the RE level are different from those addressed at the Design and Implementation levels, we believe that a concept of agent for RE that is ontologically distinct from those in Design and Implementation is needed.

3.2 Equality among agents?

Another ontological question is:

> Should all agents be "first-class citizens"? i.e., should they all have the same expressive power, or should some be treated in a more privileged fashion?

In F3, the target system is given a richer representation, in terms of models for functional and non-functional requirements. The other actors do not have these models, so that one could not easily express the "redesign" of these actors during the RE process. In CSD, the target system to be constructed is treated on a par with other agents in its environment. They are all components of the composite system.

In ***i****, all agents are equal, because intentional relationships are among agents in the model. Agents have requirements (functional and non-functional, the latter appearing as softgoals) on each other. A multi-lateral characterization of requirements relationships is more suited to a networked computing environment.

3.3 Agents and Goals

The concept of goal has been found to be a very useful one in RE, where one wants to deal with choice, and constraints on a space of possibilities (e.g., [9]). Since it is possible to have agents without goals in an RE framework (e.g., Albert), and also goals without agents (e.g., the NFR framework [4]), an important question is:

> How can agent concepts and goal concepts complement each other in an RE framework?

In several frameworks, the goal concept dominates over the agent concept, i.e., the framework is more (centrally) goal-driven than agent-driven. The agents are more on the "receiving end" of the RE process. They get responsibilities assigned to them, but they do not have much say over what they get, since they do not have the benefit of a full-fledged goal-based reasoning process in order to participate in the give and take of responsibilities.

In F3, objectives are global organizational business goals, and are separate from the Actors model. CSD is also goal-driven, beginning with global goals that are independent of agents. Agents come into the picture only after goals have been sufficiently reduced. KAOS follows the CSD philosophy, although the metamodel does allow agents to have "wishes", which can be complementary or conflicting, therefore it seems to allow agent to participate in goal negotiation from the start.

In $\boldsymbol{i}^*$, goals always belong to agents. There are no global goals (hence the name $\boldsymbol{i}^*$, which stands for distributed intentionality). Furthermore, goals appear in the structural relationships among agents, in contrast to the usual input-output relationships in most modelling frameworks. During the RE process, the goal-based relationships among agents are renegotiated. This may result in the shifting of responsibilities from one agent to another, or the splitting and merging of agents (hence redefining the identities and (logical) boundaries of agents).

3.4 Agents and Quality Requirements

Quality, or non-functional, requirements such as accuracy, usability, performance, security, costs, reliability, etc., must be taken into account fully during RE. However, unlike functional requirements, quality requirements are usually harder to deal with because they are not easily formalized. In goal-oriented RE frameworks, one cannot easily treat functional and non-functional requirements in the same way as goals because the classical treatment of goals assume that there is a clear-cut true/false criteria of success.

For example, in CSD, since the goal reduction process is formal, non-functional requirements such as costs, reliability, etc., are handled at agent assignment time, using separately attached mechanisms. These are used as evaluation criteria to decide whether a given agent can meet the responsibilities, but not as goals that are systematically reduced.

The approach is similar in KAOS. However, since global goals can initially be informal (i.e., not have a formal definition), non-functional requirements can be mixed in with functional requirements, and can be traded off among agents. However, in order for goals to be assigned as responsibilities to agents, they must be reduced sufficiently as to be expressible formally in terms of logical formulas.

In the NFR framework [4], quality goals are treated systematically using a concept of satisficing. $\boldsymbol{i}^*$ follows the NFR framework approach, bringing non-functional goals (called softgoals) into an agent-oriented framework. Softgoals, like other goals, appear as relationships among agents. Therefore quality requirements, like other requirements, are relationships among agents (as opposed to relationships with some external global designer). During redesign, quality requirements are renegotiated between agents. The $\boldsymbol{i}^*$ framework therefore offers a distributed treatment of quality issues. This is one of the areas that needs to be further developed. The WinWin framework [1] links quality goals to stakeholders, and is therefore a step in this direction.

3.5 Agents and Process

The concept of process is pervasive in systems development and in RE (e.g., [16]). When there are multiple agents, one needs to have ways of characterizing the "who" that is involved in a process, and the extent and nature of involvement. Again, we are interested in a notion of "who" that is appropriate for the RE-level, so that a physical notion of agent may not be adequate.

One can distinguish a number of levels or degrees of participation by agents in the RE process:

1. Passive involvement – An agent is said to perform certain actions or activities. This may be a description, or a prescription, as in the assignment of responsibility. The agent is not participating as an active intentional agent. The agent has no choice.
2. Constrained behaviour – An agent's behaviour is constrained by specification, within a space of possible behaviour (e.g., Albert).
3. Design choice – Agents are actively involved in making choices, and these choices and their criteria are explicitly captured by the modeller (e.g., KAOS).
4. Design choice and autonomy – Agents are actively involved in making choices, and the choices and criteria are only partially available to the modeller (e.g., $\boldsymbol{i}^*$)

There are also different kinds or "levels" of process:

1. the operational process – This refers to the processes that are executed by agents during "run-time", in the "usage-world" [15]. This is the "business process" in the target environment. An example is the "lives" of agents in Albert.
2. the design process – This refers to the processes during RE used to decide what the operational process shall be. This is not to be confused with processes in the Design phase of system development. An example is the goal decomposition process in CSD or KAOS.

3. the methodology – This refers to the collection of coarse-grained process steps that make up the RE methodology. It prescribes how the different kinds of RE activities interact, and in what sequence. For example, in CSD, one starts with global goals.

In i^*, the boundary between the operational process and the design process is not methodologically pre-determined. Any part of an operational process can be "reflected up" into the design process. The operational process, as described by a Strategic Dependency model, may still contain goals. In contrast, in most other approaches, the operational process is described in terms of actions or activities (i.e., all goals have been fully reduced). The i^* approach allows agents in the domain to decide the degree to which goals need to be resolved during the design process, leaving unresolved goals to be addressed during the operational process.

Given these various approaches to the treatment of relationships between agents and process (behaviour), and between action and reflection, there is much room for further investigation.

A related area of much interest in RE is that of scenarios. A scenario can be seen as a single instance of a process. How agents participate in scenarios – to what degree, and in what combination of action and reflection – are also important issues to investigate in an agent-oriented RE framework.

3.6 Agents and Objects

In order to do modelling, one needs support for dealing with a large body of knowledge involving numerous objects. Conceptual modelling frameworks such as Telos [15] provide knowledge structuring mechanisms such as classification, generalization, aggregation, and time to help organize knowledge along various dimensions.

How agent concepts interact with these dimensions needs to be further investigated, given that the intentional dimension of agents may bring different ontological and epistemological assumptions. For example, most frameworks treat agents as specializations of objects. However, if agents are intentional, the implications of inheriting intentional properties is not so clear.

3.7 Agents and Change

In many environments, it will be important to be able to deal with ongoing evolution [6]. The agent concept in RE may provide a pivotal point for organizing knowledge in the presence of change, because agents are often the originators of change [11]. However, it is not clear which is more stable: processes, global goals, agent goals, agent identity, or variants of agents such as roles, positions, or materialized agents? Questions of stability are also relevant when considering potential for reuse.

3.8 Scalability

In realistic settings, the number of agents that are potentially relevant could be very large, or, in the case of network computing, practically infinite. In frameworks that rely on the use of global goals, one needs to have some criteria for delineating the global system before applying the framework.

In $\boldsymbol{i}^*$, all goals are goals of particular actors (although an actor can be an aggregate such as an organization), and their implications are assessed via propagation over a network of dependencies. There is no reliance on predefined global boundaries. The premise is that the scoping criteria need to be contingent on the strategic interests of the actors, and therefore cannot be stated *a priori*. The scope of relevance needs to be determined via an analysis of the strategic dependencies of the actors. One implication of this is that the framework allows the scope of relevance to be different for each alternative solution considered during the RE process.

3.9 A single unified agent concept for RE?

In many of the questions raised in the above discussions, it is not clear that there should be a single best answer. Indeed one may need different notions of agent for different parts of RE. The contrast between $\boldsymbol{i}^*$ and Albert, for example, suggests that there may be a natural division of labour. In focusing on different parts of RE, namely the early-phase (strategic reasoning) and late-phase (specification) respectively, the two frameworks came up with substantially different notions of agent. A loose coupling between these two frameworks each with their own notion of agent may be an appropriate solution [19, 20].

4 Conclusions

We argued for the further development of agent concepts for RE. Although a number of RE frameworks have employed concepts of agents, the list of issues raised in this paper suggests that a lot more work needs to be done in order to fully exploit the potentials of introducing a concept of agent that adequately reflects the many needs of RE. The above list of issues is by no means exhaustive, as there are many other agent-related concepts, e.g., viewpoints and perspectives, negotiation support, and various notions of conflict or inconsistency. Through this discussion, we hope to stimulate further interest in agent-oriented RE, and to encourage research to address the issues raised, so as to meet the RE challenges brought about by new environments such as network-centric computing.

Acknowledgements. Financial support from the Natural Sciences and Engineering Research Council of Canada and the Information Technology Research Centre of Ontario are gratefully acknowledged.

References

1. B. Boehm, H. In, Identifying Quality-Requirement Conflicts, *IEEE Software*, pp. 25-35, March 1996.
2. J. A. Bubenko, Information Modeling in the Context of System Development, *Proc. IFIP,* pp. 395-411, 1980.
3. 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.
4. K. L. Chung, *Representing and Using Non-Functional Requirements for Information System Development: A Process-Oriented Approach*, Ph.D. Thesis, also Tech. Rpt. DKBS-TR-93-1, Dept. of Comp. Sci., Univ. of Toronto, June 1993.
5. A. Dardenne, A. van Lamsweerde and S. Fickas, Goal-Directed Requirements Acquisition, *Science of Computer Programming*, 20, pp. 3-50, 1993.
6. G. De Michelis, E. Dubois, M. Jarke, F. Matthes, J. Mylopoulos, K. Pohl, J. Schmidt, C. Woo, and E. Yu, "Cooperative Information Systems: A Manifesto," to appear in *Cooperative Information Systems: Trends and Directions*, M. Papazoglou and H. Schlageter, eds., Academic Press, 1997.

7. E. Dubois, Ph. Du Bois, F. Dubru, and M. Petit, "Agent-Oriented Requirements Engineering: A Case Study Using the Albert Language", *Proc. 4th Int. Working Conference on Dynamic Modelling and Information System - DYNMOD-IV*, Noordwijkerhoud (The Netherlands), September 1994. Delft University Press, 1994.
8. Ph. Du Bois, *The Albert II Language – On the Design and the Use of a Formal Specification Language for Requirements Analysis*, Ph.D. Thesis, Department of Computer Science, University of Namur, 1995.
9. M. S. Feather, Language Support for the Specification and Development of Composite Systems, *ACM Trans. Prog. Lang. and Sys.* 9 (2), pp. 198-234, April 1987.
10. M. S. Feather, Composite System Design, *ICSE-16 Workshop on Research Issues in the Intersection Between Software Engineering and Artificial Intelligence*, International Conference on Software Engineering, Sorrento, Italy, May 16-20, 1994.
11. O.C.Z. Gotel and A.C.W. Finkelstein, An Analysis of the Requirements Traceability Problem, *Proc. IEEE Int. Conf. on Requirements Engineering*, Colorado Springs, pp. 94-101, April 1994.
12. S. J. Greenspan, J. Mylopoulos, and A. Borgida, Capturing More World Knowledge in the Requirements Specification, *Proc. Int. Conf. on Software Eng.*, Tokyo, 1982.
13. M. Hamilton, The Shift to Net-Centric Computing *IEEE Computer*, pp. 31-39, Aug. 1996.
14. M. Jackson, *System Development*, Prentice-Hall, 1983.
15. J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis, Telos: Representing Knowledge about Information Systems, *ACM Trans. Info. Sys.*, 8 (4), pp. 325-362, 1990.
16. K. Pohl, *Process-Centered Requirements Engineering.* Wiley/Research Studies Press, New York, 1996.
17. E. Yourdon, Java, the Web, and Software Development, *IEEE Computer*, pp. 25-30, August 1996.
18. E. Yu, *Modelling Strategic Relationships for Process Reengineering*, Ph.D. thesis, Dept. of Computer Science, University of Toronto, 1995.
19. E. Yu, P. Du Bois, E. Dubois, J. Mylopoulos, "From Organization Models to System Requirements – A 'Cooperating Agents' Approach," *Proc. 3rd Int. Conf. on Cooperative Information Systems (CoopIS-95)*, Vienna, Austria, pp. 194-204, May 1995.
20. E. Yu, "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering," *Proc. IEEE Int. Symp. Requirements Engineering*, Annapolis, Maryland, pp. 226-235, January 1997.

This article was processed using the LaTeX macro package with LLNCS style