CEUR Proceedings of the 5th International i Workshop (iStar 2011)*

Requirements Engineering for Social Applications

Amit K. Chopra and Paolo Giorgini

University of Trento, Italy
chopra@disi.unitn.it, giorgini@disi.unitn.it

Abstract. We characterize social applications as those involving interaction among multiple autonomous agents. We are interested in the essential concepts and approaches for modeling such applications. We make the case that i* has some limitations with respect to the modeling of social applications. The problem is in the intentional nature of i*. The deeper roots though lie in the centralized machine-oriented approach of current requirements engineering approaches. We recommend an interaction-oriented approach to requirements modeling, modeling in terms of social commitments rather than dependencies, and in general, accommodating a distributed perspective right from the earliest phases of software engineering. For clarity, we also distinguish social commitments from various similar-sounding notions in the literature.

1 Introduction

Many of the applications that we use are social in the sense that they involve communication among two or more social agents. Banking, healthcare, e-business, emergency services, and meeting scheduling are in this sense social applications.

Let us say we want to design a meeting scheduling application. Let us consider that any meeting scheduling enactment involves agents that play convener, scheduler, and participant. There are two ways to approach the design.

In one approach, you consider a set of requirements and build a machine that meets the requirements. This is the approach one would apply when specifying a washing machine, an LCD monitor, a gate controller, or an aircraft's fly-by-wire controls. In other words, this is the approach we use to design technical artifacts. The convener, scheduler, and the participants are still social agents, but from the perspective of the machine they are merely *users*.

In the other approach, you specify (or choose or compose from existing ones) a *protocol* that supports your goal of scheduling meetings. A protocol is a specification of interaction specified with reference to roles that social agents may adopt. The meeting scheduler protocol would have the roles convener, scheduler, and assistant. It does not matter which particular social agent adopts the role. Nor does it matter what its goals are, if any. This approach preserves the original social nature of the application. *Designing a system with new requirements means designing a protocol that meets those requirements*.

We term the former approach *machine-oriented* and the latter *interaction-oriented*. The latter is about the design of protocols. In fact, the system is in a sense the protocol. There is the separate question though about the design of the participants in a protocol.

Suppose you are a social agent who notices that some other social agents have adopted the roles of convener and participant, but are waiting for someone to adopt the role of scheduler. You are willing to play the role provided some of your requirements are met by participating in the system (for example, payment for scheduling services). You do not know the design or motivations of the other participants, but what you can do is check if the meeting scheduling system, that is, the protocol, supports your requirements. Further, you may design a software artifact that encodes part of your decision-making, interacts accordingly with the other participants, and in effect represents you in the system. For the purposes of this paper, we refer to this artifact as a social agent's *surrogate*.

Interaction-orientation preserves the nature of the social application. It accommodates both the design and runtime autonomy of agents. Machine-orientation does not.

The allusion to current practices in requirements engineering would not be lost upon anyone. Current practices in requirements engineering are machine-oriented. The essential idea is to come up with the specification of a *machine* that along with the domain assumptions satisfies the requirements of the stakeholders. RE emphasizes two kinds of systems: *system-as-is* and *system-to-be*. The former is the system as it exists without the machine. It helps in understanding the environment in which the machine will be introduced. The latter is what the system will be when the machine is introduced. Presumably, in the system-to-be, the stakeholders' requirements are satisfied.

2 Objectives

Our broad objective is to understand and improve upon the software engineering of social applications. Two questions follow naturally from the discussion above.

1. What are the methodologies for designing protocols?
2. What are the methodologies for designing surrogates?

Our immediate objective though is to understand whether i* [17] is suitable (in terms of concepts) for the modeling of social applications. i* has had considerable success in the requirements engineering community, and researchers are applying it to model all kinds of applications. Obviously, the question turns on whether i* can be used to model interactions. To do this, in the following section, we outline the i* methodology, and then analyze some of its critical constructs.

3 Scientific Contributions

The idea of interaction-oriented programming traces it roots to Singh [11]. It is an idea distinct from that of agent-oriented software engineering, where the focus is on creating intelligent agents. As stated earlier, interaction-orientation says nothing about any particular agent's rationale or the design of its surrogate or any other internal information system [13].

There has been considerable progress in methodologies for designing protocols, especially as concerns the nature of protocol specifications. A great leap here came

with the idea that protocols ought to specify the meaning of communication, not its flow [12]. Social commitments have emerged as a key element of meaning. Further, it is broadly accepted that protocols cannot be formalized in terms of the goals or intentions of agents [1]. Methodologies for protocol specification have also received attention [8, 4].

The second question has received less attention in the software engineering community. In some recent work, we have shown how an individual agent may reason about his goals in light of his potential communications, specifically the social commitments he can potentially be involved in [3, 2]. In further work, we have extended this idea to address adaptation in social applications [7]. The key idea there is that it is not systems that adapt. After all, the system is nothing but a protocol. Instead it is the social agents (or their surrogates) that adapt, each autonomously from other agents in the system.

3.1 The Nature of i*

At the basic level, i* follows the standard RE conceptualization sketched above. Indeed, i* is a machine-oriented methodology; it is not interaction-oriented. Where i* differs is in the explicit modeling of organizational agents (*actor* in i* terms) and their requirements. The requirements themselves are understood in terms of the goals of the agents. The agents are related to each other by means of intentional dependencies.

The notion of agent within i* seems to be a broad one. A stakeholder could be agent; so could a legacy system; so could a service. The core idea is to model the relevant aspects of the environment via the notion of agents. The system-as-is is modeled in i* as a network of the existing agents with the intentional dependencies among them as the links. The system-to-be would introduce a machine, also an agent, towards satisfying the requirements in a suitably modified network.

The notion of intentional dependency in i*, a central one, deserves special attention. (We talk about goal dependencies but the discussion applies to other kinds of i* dependencies as well.) An i* goal dependency between two agents x and y for some goal p means that x *wants* p, and y is *able* to achieve p and in addition *intends* to achieve p.

For example, in i* one would say that the convener depends on the scheduler for its goal that the meeting be scheduled. This dependency means that (1) the convener wants the meeting to be scheduled, (2) the scheduler is able to schedule the meeting, and (3) the scheduler *intends* to schedule the meeting. Equivalently, in i* terminology, one cay say that the convener has *delegated* its goal scheduled to the scheduler.

It is interesting to contrast the notion of goal dependency with the notion of social commitment. A social commitment $\mathsf{C}(x, y, p, q)$ means that x is committed to y for q if p holds. It doesn't mean that y wants q or x wants p. Nor does it say anything about the ability of the agents to deliver p or q. Nor does it say anything about any agent's intentions. A social commitment only comes about due to interaction among the agents. This is fundamentally different from the notion of a goal dependency, as discussed above. Notably, *intends to* in i* is stated in terms of an *internal commitment*. However, internal commitment is not the same as social commitment [10]. The interaction-oriented approach talks only about social commitments.

Consider that based on the requirements analysis done with the help of the i* strategic dependency and strategic rationale diagrams, a machine for scheduling is deployed.

Later, the organization finds that the maintenance of the scheduler is too costly and therefore decides to outsource meeting scheduling to an external organization. In such a situation, one would not have information about the external organization's goals, intentions, or abilities. Hence, all the analysis done earlier with the in-house meeting scheduler would come to naught.

Consider instead that an organization went about meeting scheduling in an interaction-oriented way. First, it selected an industry-standard meeting scheduling protocol. Then it created a surrogate to play the role of the scheduler. Later it found that the surrogate was too difficult to maintain. So it outsources the scheduling to a service provider that had advertised itself as following the protocol. The interaction-oriented approach naturally supports such substitution.

3.2 Social Commitments

We have found that the notion of social commitments we refer to is often confused with other notions in the literature. We take the opportunity here to distinguish social commitments from related concepts in the literature.

Internal Commitment Cohen and Levesque's work [6] formulated a rich theory of rational action based on the concepts of *intention* and *internal commitment* to intention. Broadly, the idea is that for an agent to succeed with its intentions, it must be internally committed to realizing them. Internal commitment refers to an agent's psychological entrenchment.

Obligations Commitments are not obligations. Commitments come about only due to communication *and* hold irrespective of what exists in the agents' internal states. This is the essence of a public semantics of communication. The representation of a commitment may appear similar to that of an obligation; however, that is where the similarity ends. Obligations, as studied in the literature, represent a mental concept. For example, just because $\mathsf{C}(Barbara, Alice, paid, deliverPhone)$ does not imply $\mathsf{O}(Barbara, Alice, paid, deliverPhone)$. Neither does the implication hold in the other direction. Obligations are in the spirit of internal commitment.

Responsibilities The notion of responsibility has been applied toward the modeling of sociotechnical systems [15, 14]. The idea is that if one knows an agent's responsibilities, one can derive its requirements. However, responsibility is modeled as something that comes about because of the passing of an obligation from one agent to another via delegation. For example, a professor can delegate his obligation of going to a meeting to a Ph.D. student. The student then becomes responsible for fulfilling that obligation. However, tying responsibility to delegation definitionally unnecessarily limits the kind of responsibilities that can be captured. When an agent makes a social commitment, he is socially accountable to the creditor for its fulfillment. There is no delegation involved here, but there still is a sense of responsibility. Further, keeping an agent's autonomy in mind, only an agent itself can create social commitments. This is not to say we rule out the notion of delegation. In general, for delegation to succeed, other commitments must first be set up; for example, the delegatee must have a prior commitment to the delegator for honoring delegations [5].

Norms We understand norms as (public) conventions. Social commitments are doubly normative. One, protocols are conventions: they specify the commitments that would arise from the agents' communications. Two, commitments form the basis of compliance checking. A violation of a commitment may represent a serious exception, and creditors and the contextual community may request sanctions on the offending debtor for the violation.

4 Conclusions and Future Work

In the current paper, we advance the theme we have been pursuing for two years, but with more details specific to i*. We discussed the main limitations with respect to engineering social applications. The intentional approach of i* works for the traditional RE setting where the idea is to design a centralized machine that meets the stakeholders' requirements. However, an intentional approach is an integration-based approach. Social applications, on the other hand, necessarily have to adopt an interoperation-based approach because there is no central machine.

The distinction between mentalist (cognitive) and social notions is worth pointing out. When one talks about agents, concepts such as goals, beliefs, intentions, strategies, plans, and so on are mentalist concepts. Communication, convention, and social commitments are social concepts. When one models a system of social agents using mentalist concepts, the results are vastly inferior to when one models the same system using social concepts [12]. i* models rely exclusively on mentalist concepts (Yu uses the term *intentional*). Nowhere does communication come into the picture. Therefore, it is not clear in which sense i* would be a social approach [16]. Some other recent work [9] also fails to make the distinction between mentalist and social.

Lately there has been some work talking about the unmanageability of i* diagrams. i* has some modularity via the notion of agents; but the problem is there is no encapsulation. Dependencies do not stop at agent boundaries; they connect agents internals. In social systems, the basic assumption is that agent internals are not known. This implies encapsulation is not broken. Social commitments help us reason about such well-encapsulated agents. Social commitments would greatly enhance the manageability of specifications.

In general, RE must move away from the centralized machine-oriented perspective to a more distributed perspective that conceptualizes a system as being constituted from independently designed agents. The methodological ingredients of such an approach is the direction of research we are currently pursuing.

Acknowledgments. Numerous discussions with Munindar Singh, Fabiano Dalpiaz, and John Mylopoulos over the past two years led to the ideas in the paper. Amit Chopra was supported by a Marie Curie Trentino award. Paolo Giorgini was supported by the EU-FP7-IST-IP-ANIKETOS and EU-FP7-IST-NOE-NESSOS projects.

References

1. Amit K. Chopra, Alexander Artikis, Jamal Bentahar, Marco Colombetti, Frank Dignum, Nicoletta Fornara, Andrew J. I. Jones, Munindar P. Singh, and Pınar Yolum. Research di-

rections in agent commmunication. *ACM Transactions on Intelligent Systems*, 2011. To appear.
2. Amit K. Chopra, Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos. Modeling and reasoning about service-oriented applications via goals and commitments. In *Proceedings of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE)*, volume 6051 of *LNCS*, pages 113–128. Springer, 2010.
3. Amit K. Chopra, Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos. Reasoning about agents and protocols via goals and commitments. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 457–464, 2010.
4. Amit K. Chopra and Munindar P. Singh. Colaba: Collaborative design of cross-organizational business processes. In *Proceedings of the Workshop on Requirements Engineering for Systems, Services, and Systems of Systems*, 2011. to appear.
5. Amit K. Chopra and Munindar P. Singh. Specifying and applying commitment-based business patterns. In *Proceedings of the Tenth International Conference on Autonomous Agents and MultiAgent Systems*, 2011.
6. Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
7. Fabiano Dalpiaz, Amit K. Chopra, Paolo Giorgini, and John Mylopoulos. Adaptation in open systems: Giving interaction its rightful place. In *Proceedings of the 29th International Conference on Conceptual Modeling*, volume 6412 of *LNCS*, pages 31–45. Springer, 2010.
8. Nirmit Desai, Amit K. Chopra, and Munindar P. Singh. Amoeba: A methodology for modeling and evolution of cross-organizational business processes. *ACM Transactions on Software Engineering and Methodology*, 19(2):6:1–6:45, 2010.
9. Renata S. S. Guizzardi and Giancarlo Guizzardi. Applying the UFO ontology to design an agent-oriented engineering language. In *Proceedings of the 14th East European Conference on Advances in Databases and Information Systems*, pages 190–203, 2010.
10. Munindar P. Singh. Social and psychological commitments in multiagent systems. In *AAAI Fall Symposium on Knowledge and Action at Social and Organizational Levels*, pages 104–106, 1991.
11. Munindar P. Singh. Toward interaction-oriented programming. TR 96-15, Department of Computer Science, North Carolina State University, Raleigh, May 1996. Available at www4.ncsu.edu/eos/info/ dblab/www/mpsingh/ papers/mas/ iop.ps.
12. Munindar P. Singh. Agent communication languages: Rethinking the principles. *IEEE Computer*, 31(12):40–47, December 1998.
13. Munindar P. Singh and Amit K. Chopra. Programming multiagent systems without programming agents. In *Proceedings of the 7th International Workshop on Programming Multi-Agent Systems*, volume 5919 of *LNCS*, pages 1–14. Springer, 2010.
14. Ian Sommerville, Russell Lock, Tim Storer, and John Dobson. Deriving information requirements from responsibility models. In *Proceedings of the 21st International Conference on Advanced Information Systems Engineering*, pages 515–529, Amsterdam, 2009.
15. Ros Strens and John Dobson. How responsibility modelling leads to security requirements. In *Proceedings of the New Security Paradigms Workshop*, pages 143–149, 1993.
16. Eric S. Yu. Social modeling and i*. In Alexander T. Borgida, Vinay K. Chaudhri, Paolo Giorgini, and Eric S. Yu, editors, *Conceptual Modeling: Foundations and Applications*, pages 99–121. Springer-Verlag, 2009.
17. Eric S.K. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Proceedings of the Third IEEE International Symposium on Requirements Engineering*, pages 226–235, 1997.