1 Social Modeling for Requirements Engineering: An Introduction

Eric Yu, Paolo Giorgini, Neil Maiden, and John Mylopoulos

Information technology can be used in innumerable ways and has great potential for improving people's lives. Yet, designing systems that truly meet people's needs remains a considerable challenge. Every day we encounter systems that do not do what we want. Many software projects continue to fail at alarming rates. Much of this challenge is attributed to the difficulty of pinning down system requirements—effective ways to determine what people want when they initially conceive of the need for a software system.

This book offers a new way of approaching the requirements challenge. The central premise is that to arrive at system requirements, that is, to conceive what system to build, one should examine and understand the relationships among social actors. A system aims to improve the relationship that some actors have with other actors. Rather than focusing on the behavioral properties of software, as in a mechanistic system, we should raise the level of abstraction and ask how the system will advance the relationships that some actors have in relation to other actors.

This perspective leads to a rather different approach to requirements modeling and analysis, one that is based on describing and analyzing social relationships. We are then led to ask how we model and analyze the social world in order to lead to better system requirements. What concepts should be the basis for such modeling? What questions should the models help answer, and how can they lead us to systematically explore alternatives and analyze the consequences of our choices?

Traditionally, the task of the requirements analyst is to collect requirements statements from stakeholders: the customer and representatives of users. These statements say what the system should do (functionality) and at what levels of quality (nonfunctional properties such as performance, reliability, extensibility, usability, and costs). For large systems, there can be a large number of such statements coming from many stakeholders. The analyst aims to ensure that these statements are consistent (i.e., they do not contradict each other), complete (i.e., they fully reflect what the stakeholders are expecting from the system), and unambiguous (i.e., sufficiently precise so that they will not be misinterpreted by the developers).

The approach outlined in this book addresses issues that should come before the traditional requirements analysis activities. We call this early requirements engineering. We argue that in order to arrive at good requirements, one needs to understand the underlying motivations behind the proposed system. It is not enough to write down what users and customers say they want. Users and customers are often not able to articulate these wants directly. Instead, the analyst needs to help them uncover their real needs. Users are often unaware of what is possible or have misconceptions about what is feasible, especially when technology is advancing quickly. When users express their requirements in concrete terms that include technology elements, these may well be premature conceptions of solutions that do not actually respond to their real needs.

Before one can properly understand requirements, one needs to ask why the proposed system is needed, who is involved, and what relationships exist among various actors. One needs to understand how things are done under current conditions, why they work or do not work, from whose perspective, and according to what criteria. In specifying a new system, that is, the requirements, one is in effect rearranging relationships among the social actors. Experienced analysts recognize the importance of the human social dimension, and respond to people's concerns and desires. Current requirements models and techniques, however, provide support only for stating the results of such deliberations. Existing requirements models focus on behaviors and activities, and information entities and relationships among concepts. The understanding and analysis of the social dimension rely on the skills and experience of the analyst, without models or systematic analytical support.

The recognition that understanding and modeling the environment should be an integral part of requirements analysis was a major milestone for the field (Jackson, 1983; Greenspan, Borgida, & Mylopoulos, 1986). Today, it is widely accepted that the requirements for a system must be defined in relation to its intended environment. However, the relationship is focused on behavioral interactions between the system and the environment, in terms of, for example, actions and responses, or state changes. The approach in this book considers relationships at the intentional level, as relationships among social actors.

A Social Worldview for Requirements Analysis

To begin, we need to adopt a different starting point for understanding the world in which software and information systems are situated. Traditional requirements analysis adopts a mechanistic view of the world: the world consists of entities and activities that are fully knowable and predictable.

In adopting a social worldview, we see the world as having intentionality, that is, there are intents and reasons and motivations behind behavior. Intentionality originates from actors, such as human beings. Intentional actors have wants and desires. They perform actions to fulfill their wants and desires. Actors can choose what actions to take. We say that actors are autonomous, in that they have freedom to choose their actions.

Actors do not exist in isolation. They exist in some shared environment with other actors, and interact with each other. Importantly, we recognize that actors do not only interact with each other physically and behaviorally, but also relate to each other at an intentional level. Thus their interactions are not predefined sequences of actions and reactions, but are coordinated through their respective wants, desires, and commitments. Actors collaborate, cooperate, and compete with each other. In a social world, actors can be said to be only semiautonomous, because their actions take into account their relationships with each other.

New technology systems will be viewed by each actor as potentially beneficial or threatening. Conventional requirements analysis provides only models that describe the operational aspects of a technology system and its environment. If we are to seek requirements that respond to stakeholder desires and concerns, we need to view stakeholders as actors having strategic interests, not just operational involvement with the proposed system. A new system may enable someone to perform some tasks more speedily, more cheaply, or more conveniently, but may impose new burdens on some other actors. One may be able to store and manage vast amounts of information more readily while incurring potential privacy and security risks.

In a social approach to early requirements analysis, the strategic interests of stakeholders should be used to guide the search for alternative conceptions for the new system. Each actor will seek to advance its strategic interests, or at least protect them from being eroded. If some actors are significantly adversely affected, the new system, if implemented, will likely fall into disuse or, worse, be sabotaged. These consequences reflect the autonomy of actors in a social world.

One consequence of the autonomy premise is that the analyst does not have privileged or unlimited access to the internal mental states of these actors. Unlike actions or behaviors that are observable, intentional properties can only be inferred. In general, we say that intentional properties of an actor are externally attributed by the analyst, based on information obtained by indirect means. One can never be certain of the validity of these attributions.

The intentionality premise is adopted here for pragmatic reasons, rather than as a philosophical position about the true nature of the world. An intentional characterization of social actors is used to facilitate analysis leading to requirements, leaving aside debates about whether any particular actor is truly intentional or not. This approach allows us to use the same notion of intentional actor to encompass actors that contain combinations of human and nonhuman elements in varying degrees.

Modeling

Why should social modeling be part of an engineering approach? A central ingredient of any engineering method is the use of appropriate models. Models provide abstractions for describing, understanding, and analyzing a complex world. By adopting a certain set of abstractions, a model highlights selected aspects for attention while omitting others. Traditional requirements models focus on the mechanistic aspects of systems. We argue that early requirements engineering needs to focus on the social dimension of systems and their environments. The need to understand the human social dimension in order to design effective systems is of course not a novel idea. Since the early days of computing, there have been well-established academic research streams that have studied the social impacts of computing, the behavioral aspects of computer use, and so on. What we aim to offer is a modeling approach that is part of an engineering method that provides systematic techniques and tools that can provide smooth linkages to the rest of the system development process, including technical system design and implementation.

As with other modeling techniques, in choosing to highlight certain features of a complex reality, many other aspects are omitted. The complexity of the social world presents a formidable challenge for a modeling approach. Any model is necessarily reductionist and will have its blind spots. In adopting a model, one needs to be constantly aware of its limitations. However, we hope that even a modest attempt at bringing social modeling and analysis directly into mainstream requirements engineering will be an advance over the traditional, almost exclusive, focus on the mechanistic aspects of system requirements.

Goal-Oriented Requirements Engineering

The social, agent-oriented approach builds upon goal-oriented techniques that have been widely studied in requirements engineering research in recent years. The relation between a goal—a condition to be achieved—and the means for achieving that goal has not been exploited much in traditional requirements engineering. Popular modeling languages and notations such as UML (Object Management Group, 2009) and earlier ones, such as Data Flow Diagrams (DeMarco, 1979) or SADT (Ross, 1977), use decomposition as the main abstraction mechanism for incrementally revealing detail. The decomposition mechanism, however, does not support the representation or analysis of alternative ways for achieving a goal. The languages RML (Greenspan et al., 1986) and Telos (Mylopoulos, Borgida, Jarke, & Koubarakis, 1990) recognize assertions as an ontological category on a par with activities and entities, but do not explicitly link activities to assertions through a means-ends or goal-achievement relationship.

Feather (1987) first outlined a goal-oriented requirements framework. Subsequent frameworks in which goals play a central role include the NFR framework (Chung, Nixon, Yu, & Mylopoulos, 2000), KAOS (van Lamsweerde, 2001), and GBRAM (Antón, 1996). In emphasizing the use of "why" questions, goal modeling can help in requirements elicitation. By navigating up and down the means-ends hierarchy, one can determine whether a requirement is overspecifying, that is, treating a means as if it were an end in itself. Goal modeling can improve completeness of requirements. By making goals explicit, one may uncover missing elements that are needed in addition to already identified requirements. Goals provide a richer context for understanding and interpreting requirements, because they relate at a higher level to the business or application domain. Indeed, higher-level goals may be far removed from any technology solution. A chain of means-ends links, however, will provide the reasoning and rationale that connect business objectives to various technology options. The means-ends analysis will also identify components in the environment (human or other technology systems) that are needed to work with a proposed solution.

A goal analysis reveals conflicting desires or expectations, thus allowing trade-offs to be explicitly modeled and managed. A goal model can help manage change, because many changes can be recognized as a change to the means for achieving an unchanged goal. Making goals explicit therefore provides stable reference points for interpreting and managing change. The means-ends dimension also provides a path toward design, because technology solutions are means for achieving ends in the business domain. Goals provide criteria and guides for generating and evaluating potential solutions.

From Goal Orientation to Agent Orientation[1]

The agent-oriented approach takes advantage of the strengths of goal orientation. However, it also recognizes that goals originate from many different actors, and that the relationships among the actors should be a crucial part of the worldview. In most goal-oriented requirements frameworks, all goals are treated from the single viewpoint of the requirements analyst. In the agent-oriented approach, as exemplified in *i**, goals and other intentional properties are attributed to various actors. The alternatives that are explicitly represented in goal models are now viewed as choices that can be exercised by the actor being modeled, not as choices to be made by the requirements analyst. The actor has freedom in making these choices: the idea of actor autonomy.

Interactions among goals within an actor are treated differently from those that cross actor boundaries. Actors do not have direct access to the intentional content of other actors. Relationships among actors are modeled as dependencies. This approach recognizes the autonomy of actors. Vulnerabilities exist because dependencies can fail—for instance, when a dependee does not deliver a dependum to a depender.

In analyzing competing goals, we note that there are trade-offs within the scope of an actor. But it is also important to recognize how gains for one actor may be losses for other actors.

The Changing Needs of Requirements Engineering

In adopting a social modeling perspective, we recognize that goal-oriented requirements engineering techniques by themselves are not sufficient. In today's networked world, automated systems are increasingly made up of semiautonomous interacting units, often operating under separate administrative control or ownership. The human systems in which the automated systems are embedded are also increasingly adopting network configurations, consisting of loosely coupled, locally empowered individuals and units. Recent directions such as e-business model innovations, service orientation, open source, Web 2.0 and social networking, software as a service, and global development teams are indicative of this overall trend.

A goal-oriented approach that treats the systems and the social environment as a network of interacting goals is inadequate because it is critical to attribute goals to different actors. Furthermore, analysts have only imperfect access to these actors. We need to recognize that the intentional models will be incomplete and potentially inaccurate, to different extents for each actor. In an already heavily networked world, any new system will be interacting with many existing actors over which the designer of the new system will have limited knowledge and control.

Open Research Issues

The chapters in this book represent some initial steps in this new modeling approach for requirements engineering. The framework should be viewed as one particular conception of an agent-oriented, social approach to requirements modeling. The research work represented in this volume provides an illustration of how this conception can be applied, adapted, and extended in various directions.

In reviewing the conception of the social world as outlined in the premises earlier, one can conclude that the conception is rather sketchy and simplistic. Indeed, much more work is needed to refine or explore alternative conceptions that will meet the practical needs of requirements analysts and engineers. For example, what are the desirable notions of actor autonomy? How can we model cooperation and joint activities better? Are conflicts and competing interests easy to represent and analyze? What notions of ownership, power and control, and trust are appropriate? What abstraction mechanisms can be used to better support large-scale models? How should intentional properties be related to behavior? Does the current conception provide effective linkages to other representations, such as those for plans and scenarios, and process models? What notions of time should be incorporated in social models? How do we deal with importance and priorities? How well does the social modeling approach lead to functional and nonfunctional quality requirements? How can we incorporate quantitative reasoning? Are viewpoints and perspectives of different actors sufficiently represented and supported in analysis? How do we ascribe identity and individuation to actors, given that the actor notion can be an abstraction not necessarily tied to concrete physical reality? Some of these questions are addressed in chapters in this volume, but all of them deserve further exploration.

Bringing Change to Requirements Practices

Social modeling is not found or supported in most requirements processes and methods in use today. If social modeling in requirements projects is to be widely taken up by analysts, there is a need to fit it to existing processes, methods, and tools that organizations have invested a lot of time and money in. Social modeling techniques will need to integrate with and complement existing techniques such as use cases. For example, it might still be necessary to report the outcomes of social analyses, in the form of requirements statements, on different actors. We still need to understand how to embed social modeling in established requirements approaches.

Furthermore, projects wanting to use social modeling will need evidence that these new approaches can both scale to analyze large-scale models, and deliver new and valuable insights to analysts. Producing and analyzing models that scale depends in part on new software tools that are simple and easy to use. Empirical evidence that social modeling can deliver value to projects is needed. Some chapters in this volume report early evidence, but more is needed.

Structure of This Book

This book is organized into five sections. The first section consists of this introductory chapter and the doctoral dissertation of Eric Yu, which introduced the *i** framework. Although much new work has been done since, the original dissertation continues to be a useful reference. The remaining four sections offer samples of research that apply, adapt, extend, or evaluate the social modeling concepts and approach originally proposed in the *i** framework.

Part II includes four chapters on applications and experiences, covering diverse areas from air traffic management to organizational networks, to business processes, and knowledge management. Four chapters in part III provide illustrations of different approaches to security and privacy, an area in which social modeling is considered to have particular potential. In part IV, the use of social modeling in the context of software development is explored. Finally, part V includes four chapters that illustrate extensions to the *i** framework, as well as its evaluation.

Note

1. The term *agent* is used here in a broad sense, interchangeably with the term *social actor*, and can encompass human as well as technology systems. In frameworks such as *i** and Tropos, specific distinctions are made between actors and agents.

References

Antón, A.I. (1996). Goal-based requirements analysis. In *Proceedings of the 2nd IEEE International Conference on Requirements Engineering* [RE'96] (pp. 136–144). Los Alamitos, CA: IEEE Computer Society Press.

Chung, L., Nixon, B.A., Yu, E., & Mylopoulos, J. (2000). *Non-functional Requirements in Software Engineering.* Norwell, MA: Kluwer Academic.

DeMarco, T. (1979). *Structured Analysis and System Specification.* New York: Yourdon Press.

Feather, M.S. (1987). Language support for the specification and development of composite systems. *ACM Transactions on Programming Languages and Systems, 9*(2), 198–234.

Greenspan, S.J., Borgida, A., & Mylopoulos, J. (1986). A requirements modeling language and its logic. *Information Systems, 11*(1), 9–23.

Jackson, M.A. (1983). *System Development*. Upper Saddle River, NJ: Prentice-Hall.

Mylopoulos, J., Borgida, A., Jarke, M., & Koubarakis, M. (1990). Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems, 8*(4), 325–362.

Object Management Group (OMG). (2009). *UML resource page.* Available at http://www.uml.org/.

Ross, Douglas T. (1977). Structured Analysis (SA): A language for communicating ideas. *IEEE Transactions on Software Engineering, 3*(1), 16–34.

Van Lamsweerde, A. (2001). Goal-oriented requirements engineering: A guided tour. In *Proceedings of the 5th IEEE International Symposium on Requirements Engineering* [RE'01] (pp. 249–263). Los Alamitos, CA: IEEE Computer Society Press.