Social Modeling and *i**

Eric S. Yu

Faculty of Information, University of Toronto
Toronto, Canada M5S 3G6

Abstract. Many different types of models are used in various scientific and engineering fields, reflecting the subject matter and the kinds of understanding that is sought in each field. Conceptual modeling techniques in software and information systems engineering have in the past focused mainly on describing and analyzing behaviours and structures that are implementable in software. As software systems become ever more complex and densely intertwined with the human social environment, we need models that reflect the social characteristics of complex systems. This chapter reviews the approach taken by the *i** framework, highlights its application in several areas, and outlines some open research issues.

1 Why Social Modeling

In many scientific and engineering disciplines, the principles, premises, and objectives of the field are embedded in and manifested through the models that are the daily conceptual tools of the profession. The models reflect the kinds of understanding that is sought by practitioners of the field. In software and information systems engineering, the dominant modeling constructs have revolved around static relationships (as in entity-relationships models and class diagrams) and dynamic and behavioural properties (as in process models and state-based formalisms). This focus is understandable as conceptual models are ultimately translated into data and operations for machine execution. For a system to be successful however, it must function within the context of its environment. As the need to model and characterize the machine's environment was increasingly recognized, these same modeling techniques and formalisms have been extended to cover the world in which the machine operates, and how the machine interacts with that world. The world was thus largely seen through the lens of the mechanistic operations of computers.

In a keynote speech in 1997, Professor John Mylopoulos identified four main classes of modeling ontologies that would be crucial "in the time of the revolution." Static and dynamic ontologies were well developed and widely adopted. Two new kinds of modeling – intentional and social – were needed to respond to the emerging needs of the information revolution [81].

Few would have predicted the way the revolution has unfolded. In 1997, the Netscape browser was still a novelty, and the world-wide web was hardly a household name. XML had not yet been introduced. Computer use, especially in the information systems area, was dominated by business applications within organizations, with trained users in a job setting. Today computer use is all but taken for granted. RFID, GPS, online banking and shopping are everywhere. New generations grow up unable to imagine life without Google, Wikipedia, Facebook, instant messaging, or texting. Back in the work world, organizations are adopting Enterprise 2.0, playing catch up with the Web 2.0 services that their employees and patrons have already taken for granted in their personal and social life. The revolution continues, with new technologies and services emerging all the time – e-book readers, location-based services, digital paper, and so on. Information technologies and systems are impacting people's lives in deeper ways than ever before. Every innovation has the potential to bring benefits, as well as threats to privacy, livelihoods, and even cherished cultural values. In principle, the possibilities for good are limitless. The concerns and risks are also very real. What methods and techniques can software and information systems professionals use to deal with these questions? Social modeling is more relevant than ever before.

The *i** modeling framework [122][123] was an attempt to introduce some aspects of social modeling and reasoning into information system engineering methods, especially at the requirements level. Unlike traditional systems analysis methods which strive to abstract away from the people aspects of systems, *i** recognizes the primacy of social actors. Actors are viewed as being intentional, i.e., they have goals, beliefs, abilities, and commitments. The analysis focuses on how well the goals of various actors are achieved given some configuration of relationships among human and system actors, and what reconfigurations of those relationships can help actors advance their strategic interests. The *i** framework has stimulated considerable interest in a socially-motivated approach to systems modeling and design, and has led to a number of extensions and adaptations.

This chapter aims to present an overview of the ideas behind the *i** framework, some of the main application areas, and discusses some possible future directions.

2 Premises and Features of *i** Modeling

From the earliest days, there have been concerns about the pervasive impacts that computing technology was having on society (e.g., [57]). The concerns included humanistic, ethical, as well as pragmatic ones – as many technically sound systems fell into disuse, or met with resistance from users [70]. Studies on the social impact of computing have raised awareness and sensitivity to the potentially negative as well as positive impacts of technology on people's lives. However, it has been difficult to make social understanding and analysis an integral part of the mainstream system development process.

The *i** modeling approach is an attempt to bring social understanding into the system engineering process by putting selected social concepts into the core of the daily activity of system analysts and designers, i.e., by adopting a social ontology for the main modeling constructs. Social analysis would not be an adjunct to technical analysis, but would be the basis for driving the entire system development.

To overcome the limitations of the mechanistic worldview, we shift our attention away from the usual focus on activities and information flows. Instead, we ask: what does each actor want? How do they achieve what they want? And who do they depend on to achieve what they want? In the following, we review each of the main premises of *i** [114][115], and discuss how they are manifested through the features of the modeling framework.

2.1 Actor Autonomy

We adopt as a premise that the social world is unknowable and uncontrollable to a large extent. From the viewpoint of conventional modeling, this may seem unintuitive and prohibiting. What is not knowable can hardly be modeled. Interestingly, this premise provides a way out of the usual mechanistic conception of the world.

In *i**, the central conceptual modeling construct is the actor. It is an abstraction which is used to refer to an active entity that is capable of independent action. Actors can be humans, hardware and software, or combinations thereof. Actors are taken to be inherently autonomous - their behaviours are not fully controllable, nor are they perfectly knowable.

This notion of autonomy is distinct from the one in artificial intelligence, where it refers to an advanced capability to be achieved by design and technological implementation. Autonomous agents in AI are artificial agents implemented in hardware and software which can act on their own without human intervention. In social modeling, we take actor autonomy to be a characteristic of the real-world social phenomena that system designers have to contend with.

2.2 Intentionality

Although the behaviour of actors are not fully knowable or controllable, they are nevertheless not completely random. To explain and characterize their behaviour, we attribute motivation and intent to actors. By modeling what actors intend to achieve, we obtain a higher level characterization without specifying their exact behaviour.

In *i** modeling, we focus on intentional properties and relationships rather than actual behaviour. By not describing behaviour directly, an intentional description offers a way to characterize actors that respects the autonomy premise. Conventional system modeling which offers only static and dynamic ontologies leads to an impoverished and mechanistic view of the world. Intentional modeling provides a richer expressiveness that is appropriate for a social conception of the world. By attributing intentionality, we can express *why* an actor undertakes certain actions, or prefers one alternative over another. An intentional ontology allows analysis of means-ends relationships and of the space of alternatives for each actor.

Various notions of actor are included in some non-intentional modeling frameworks and languages, e.g., in the form of stick figures in UML use case diagrams [85], and swim lanes in BPMN [5]. These actors are not intentional or autonomous, so are inadequate for social modeling. Recent work in goal modeling in requirements engineering (e.g., [108][96]) have developed intentional modeling ontologies, but have not emphasized the social dimension of intentionality. The name *i** stands for distributed intentionality, which puts intentionality within the context of social networks of autonomous actors.

2.3 Sociality

Social phenomena are arguably infinitely rich. The treatment that a modeling framework can provide is necessarily limited. Conceptual modeling frameworks aim to offer succinct representations of certain aspects of complex realities through a small number of modeling constructs, with the intent that they can support some kinds of analysis.

In *i**, we choose to focus on one aspect of being social – that the well-being of an actor depends on other actors. Actors depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished. By depending on some other actor, the depender actor takes advantage of opportunities that are made available through dependee actors. For example, my life is made easier by mechanics who are able and willing to repair my car, even if I myself am not capable. At the same time, as I depend on someone else, I become vulnerable to not receiving what I expect from them. These dependencies are therefore strategic to the actors concerned because they can be beneficial or harmful to their well-being. Actors would choose what dependencies to have according to their judgement on the potential gains and losses from them.

In *i**, the *Strategic Dependency (SD) model* (Fig. 1) is a network of directed dependency relationships among actors. A dependency link indicates that one actor (the *depender*) depends on another (the *dependee*) for something (the *dependum*). Four types of dependencies are distinguished. If the dependum is stated as an assertion, it is called a *goal dependency*. The depender wants the dependee to make the assertion true, without specifying how it is to be achieved. If the dependum is stated as an activity, it is called a *task dependency*. The depender wants the dependee

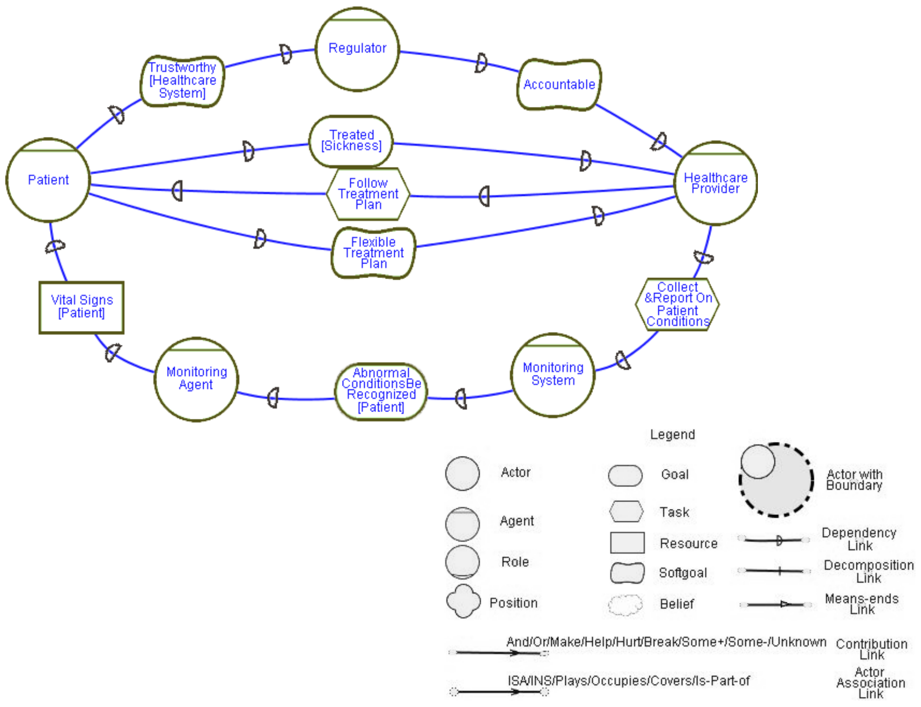

Fig. 1. A Strategic Dependency (SD) Model (from [114])

to perform the task as specified by the description of the activity. The dependency types therefore provide a way to convey the kinds of freedoms allowed in a relationship. In a goal dependency, the dependee is free to adopt any course of action, as long as the goal is achieved. The depender does not care how it is achieved. In a task dependency, the dependee's actions are confined, as stipulated by the depender. However, since the task description can seldom be complete and in full detail, the dependee still has freedom beyond what is specified.

A *resource dependency* is one in which the dependum is an entity, which can be an information or material object. The depender wants the dependee to furnish the entity so that it can be consumed as a resource, but is not concerned about the activity or any problem solving that may be needed to produce the entity.

A fourth type of dependency, called a *softgoal dependency*, is one in which the dependum is a quality, such as fast, cheap, reliable, secure, and so forth. A softgoal dependency is similar to a goal dependency except that the criteria for achievement of the quality goal is not sharply defined a priori, but may require elaboration and clarification. Consultation between depender and dependee may be required. The softgoal concept was first used to deal with non-functional requirements in software engineering [11]. It provides a useful mechanism for modeling many concepts in the social world which require contextual interpretation.

The SD model may be contrasted with process models employing dynamic ontologies. Unlike typical process models such as dataflow diagrams or activity diagrams which focus on information flow or control flow, the SD model is a higher level abstraction which depicts what actors want from each other, and the freedoms that each actor has. The SD model therefore acknowledges the autonomy of actors in a social world.

The SD model depicts external relationships among actors, while remaining silent regarding the internal makeup of the actors. For example, actors may possess knowledge that enables them to achieve goals and perform tasks, but this knowledge is not made explicit in the SD model.

Since an actor is autonomous, it may choose not to live up to the expectation from a depender. By analyzing the network of dependencies, one can infer that some dependencies are more likely to be viable than others. For example, if A depends on B for something, while B in turn has some substantive dependencies on A, directly or indirectly, then A's dependency is likely to be viable. Dependency failures may propagate along a chain of dependencies. When an actor (as dependee) is committed to delivering on a dependum, its efforts may prevent the failure from further propagating [122].

2.4 Rationality

The kinds of analyses that can be performed with the SD model are limited due to the strong assumption about actor autonomy. The SD model focuses on external relationships, while staying mute on internal makeup. In making sense of a social world, we often attribute motivations and intents to actors. We construct coherent explanations of their behaviour by relating their actions to attributed goals and motives.

In the *Strategic Rationale (SR) model* (Fig. 2), we attribute goals, tasks, resources, and softgoals to each actor, this time as internal intentional elements that the actor wants to achieve. A *means-ends link* is used to connect a task to a goal, indicating a specific way to achieve the goal. Typically there is more than one way to achieve a goal, so a goal in an SR model prompts the question – how else can this goal be achieved?

A task can be further specified through *task decomposition links* to indicate the subtasks, subgoals, resources, and softgoals that need to be performed or satisfied in order for the task to succeed.

Tasks have *contributions links* to softgoals indicating how they contribute to achieving those qualities (positively or negatively, with what strength). High level softgoals are refined into more specific softgoals through AND, OR combinations as well as partial contributions. Softgoals help distinguish among alternate tasks that can achieve the same goal, but differ in how they affect desired quality attributes. For example, the goal to Keep Well can be achieved through Patient-Centred Care or Provider-Centred Care, but the two options affect the patient's Lifestyle and Quality of Care differently. A qualitative reasoning procedure is used to propagate labels across the graph to evaluate goal achievement. When goals are not sufficiently met, actors look for further alternatives that produce more favourable outcomes.

In attributing rationality to actors in an *i** model, we are not asserting that these actors are intrinsically rational. Rather, rationality is externally attributed so that we as analysts can reason about their behaviour. In accordance with the autonomy premise, the model is inherently incomplete and may well be inaccurate.

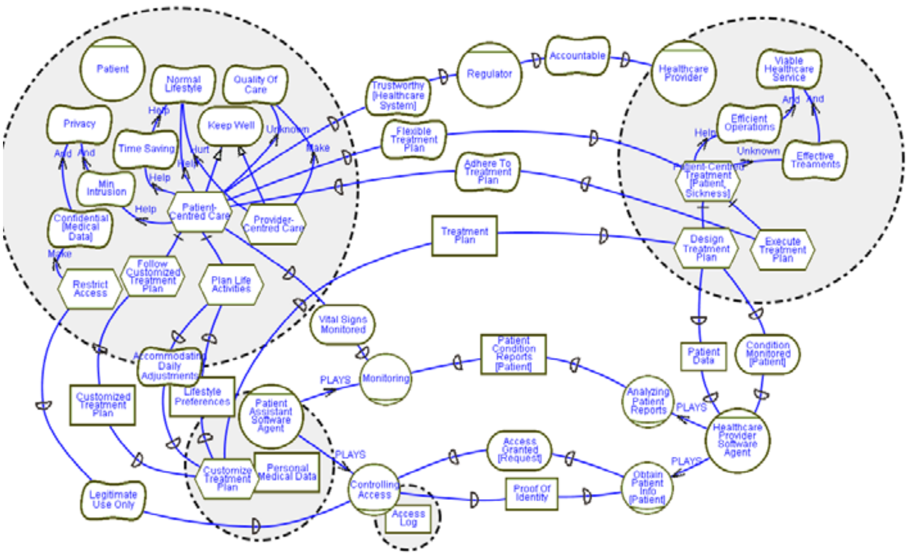

Fig. 2. A Strategic Rationale (SR) Model (from [114])

A belief is an assertion held to be true by an actor. It is useful for noting assumptions or justifications which when revoked, should trigger re-evaluation of affected decisions. Unlike a goal, the actor does not aim to make a belief come true.

2.5 Contingent Boundaries and Identities

Since we take the actor to be a modeling abstraction, its identity and scope are determined by the modeler. For example, it is up to the modeller to model persons in a work group individually as actors, or the entire group as an actor, or each person supported by software tools as an actor. Each of these would offer different opportunities for analysis. Organizations and communities acting coherently can also be treated as actors. Actors can have selfish as well as altruistic goals.

In *i**, the relationship that an actor has with other actors serves to demarcate the boundary of the actor at an intentional level. When an actor delegates a responsibility to another actor, the inter-actor boundaries shift accordingly.

Social modeling needs to deal with physically embodied actors such as humans as well as abstract "logical" actors such as roles. In *i**, the term *agent* is used to refer to actors with physical embodiment. An agent may play multiple *roles*. A set of roles typically played by one agent is called a *position*.

2.6 Strategic Reflectivity

In conventional systems analysis, the models typically provide an operational level description of the system, e.g., the information and control flows. The activities and reasoning used to improve the operation of the system, e.g., how to improve efficiency, reliability, cost, security, etc., are typically done outside of the models. An intentional ontology, such as the use of goal models (e.g., [11][108] would make the intentional dimension explicit, allowing trade-offs across multiple competing or synergistic goals.

In a social ontology, such consideration and tradeoffs would be modelled from the viewpoint of each actor. Each actor reflects upon its relationships with other actors, and makes judgements about the merits of various configurations with respect to its own strategic interests.

In *i**, each operational configuration is typically expressed through an SD model. The alternatives that are explored in an SR model refer to the alternative SD configurations that have different implications for the various strategic interests held by each actor.

3 Social Modeling for Requirements Engineering

In system development, requirements engineering (RE) serves as the crucial interface between the world of the application domain and the world of computing technology.

Much effort has been devoted to developing requirements models and languages that can lead to precise specifications of what the system should do. A major milestone was the recognition that requirements need to be defined in relation to the environment [9][48]. Modeling the world was therefore as important as modeling the machine [38]. The dominant ontologies for requirements modeling were static and dynamic ontologies, centring around entities and relationships (what exists), activities and processes (what occurs), and assertions and constraints (what holds true), e.g., [38][85].

When the social environment is complex, social modeling offers a new kind of analysis that could not be achieved within the mechanistic world view of static and dynamic ontologies. *i** modeling and variants have been used to obtain requirements in domains as diverse as air traffic control [71], agriculture [90], healthcare [56][3], environmental sensing [33], e-government [19], submarine systems [89], telecommunications [2], industrial enterprise [91], and business processes (see next section).

Each of these environments involves many social actors with disparate interests, a multiplicity of roles, and complex networks of relationships. Modeling the relationships at an intentional level offers a higher level of abstraction for analysis. Intentional modeling can also include stakeholders who have no direct information flow or activity interaction with the operational system, such as regulators, investors, or the general public, who may nevertheless have an influence over the system.

By identifying relationships among stakeholders using the SD model, weaknesses in existing relationships can be revealed. The SR model provides a systematic way for exploring the space of alternatives for achieving stakeholder goals, including different ways of using technology systems. This type of analysis is important at the early stages of system conception, but is not supported by conventional requirements modeling techniques. We refer to this as "early" requirements engineering.

Intentional social modeling can be complemented with other techniques. In the RESCUE methodology [72], *i** is used in conjunction with scenario techniques, and is synchronized at several stages to cross-check requirements obtained. Creativity workshops are also used to complement the model-based techniques. The joint use of the Goal-oriented Requirements Language (GRL, a variant of *i**) and a scenario mapping approach is also promoted in the User Requirements Notation (URN), a newly approved ITU-T standard [46][2][63].

The social modeling of *i** leverages the techniques developed in goal-oriented RE, such as systematic refinement and the eventual operationalization of goals, and goal graph evaluation [11][42]. Both the goal-oriented and social approach fall within the category of problem-oriented RE as discussed by Wieringa [112], in contrast to solution-oriented RE techniques which aim to specify the target system, e.g., IEEE 830 [45] and UML [85]. In the intentional ontology of goal-oriented RE, the problem is characterized by a set of goals, solutions are explored and evaluated as alternative ways for achieving the goals. In the social perspective taken in *i**, there is no single unified view of "the problem". Each stakeholder has strategic interests to pursue, and they have limited knowledge of and control over each other. They seek to advance their interests by considering different configurations of dependency relationships.

4 Social Modeling for Software Development

Software processes are social too. Software development is a complex social activity. Despite advances in tools, software work continues to be labour and knowledge intensive. Large numbers of people with specialized roles and skills collaborate in the development and maintenance of software products and services. Numerous approaches and methodologies have been proposed on how to organize and guide the software process, ranging from heavily disciplined, tightly controlled methods to lightweight "people-centred" agile ones [4]. Compared to other more mature engineering disciplines, software work is still highly variable and difficult to organize and manage. Schedule and budget overruns are commonplace, and high quality is hard to achieve.

Various modelling techniques have been used to support software processes (e.g., [15][84]), with special emphases on workflow support, tool automation, and method reuse. Few modeling methods, however, have focused on the social aspects of software work. Social considerations such as cooperation and conflict, motivation and rewards, responsibility and control, knowledge sharing and reuse can be critical to the success of any software project.

*i** modeling has been used to bring out the human social dimension of software processes [120]. [8] and [22] provide examples of applications of social modeling to the maintenance process in industrial settings. Knowledge management issues are analyzed using *i** in [39] and [105].

Software is social too. Having introduced social models in requirements engineering, a more radical proposal is to use social modeling in the design and analysis of software itself. As software is increasingly distributed and network-based, large software systems are taking on the characteristics of social systems - being composed of units interacting with each other with relative autonomy and on a peer-to-peer basis rather than the traditional hierarchical authority and centralized oversight. Indeed, the agent-oriented approach to software engineering is emerging as a promising new paradigm for constructing software [41].

In the past, modeling methods have typically followed paradigm shifts that originated from programming. Thus structured analysis followed structured programming and structured design [17]. Object-oriented analysis adopted concepts from object-oriented programming and object-oriented design [12]. In agent-oriented software, social characteristics are metaphorically attributed to complex software interactions. However, social modeling as envisioned in [81] and exemplified in *i** is not meant to be metaphorical. Rudimentary and reductionist as it may be, the social modeling is intended to reflect aspects of social reality, not only of artificial systems. Complex software systems are social not only because they have characteristics that resemble human social systems, but because they are actually driven by complex human social systems. A truly innovative way to reconceptualise software development is therefore to see software as in fact social. Social modeling can be used throughout the software lifecycle, not just at the requirements level.

Tropos. The Tropos project, initiated by Professor Mylopoulos, adopts the social ontology of *i** at the early requirements level. As the development process progresses, features of the social ontology are retained as much as possible during late requirements and architectural design. Detailed design and implementation are done on an agent software platform such as Jack or Jadex [7]. Software components are treated as having social characteristics. Formal Tropos combines formal techniques such as model checking with the social model analysis of *i** [25].

*i** modeling has also been used to guide COTS package selection [24], database design [49], data warehousing [88] and business intelligence [92]. An empirical evaluation of the *i** framework in a model-based software generation environment was presented in [23].

5 Social Modeling for Enterprise Engineering

Business processes. A common pitfall in enterprise IT is the adoption of technology without a clear and detailed understanding of how the technology will contribute to business goals. The business process reengineering (BPR) movement [40] has been instrumental in highlighting this pitfall. The conception of a business process served as a focal point for interaction between business analysts and IT developers. The use of models to describe and analyze business processes became a centerpiece in enterprise information systems engineering, so that the business can be understood and analyzed before considering technology solutions. Most business process modeling techniques, from flowcharts to the recent BPMN [5], were adapted from system analysis, and inherit a mechanistic world view, albeit simplified to engage business participants. Main features include information flows, activity steps, branching and merging, etc.

Business process models that show concrete flows and behaviour are easy for business stakeholders to validate, and provide good starting points for technology implementation. However the static and dynamic ontologies employed only describe what happens, but cannot be used to explain why, or to explore alternatives. Social models such as *i** can be used not only to relate business processes to business goals, but to the goals of various stakeholders who would be affected by any change (e.g., customers, employees, regulators, investors, etc.) [119][121]. Taking the interests of a full range of stakeholders into account during the redesign of a business process is likely to lead to process innovations and technology systems that are more broadly accepted and viable [56][3]. Social factors such as power and conflict, often the sources of failures, can be brought in for systematic analysis as part of the system development process. The Strategic Rationale model in *i** supports reasoning about alternate process designs and social configurations [121].

Compared to other socially motivated modeling techniques (e.g., [75]), *i** attempts a deeper social ontology [116], incorporating concepts such as strategic dependencies and actor autonomy [16]. While social analysis using narrative text can be much more nuanced and therefore cannot be replaced by modeling, a model-based approach can provide more direct and traceable linkages to system development, making such social analysis more likely to have impact on technical system design and implementation. Some methods that start from *i** models leading to business processes execution include [62] [59] [53] [111] [29] [60]. An *i**-based method for process reengineering and system specification is developed in [37].

Enterprise architecture. As information systems multiply in organizations, systematic frameworks and approaches have been proposed to manage systems not one system at a time, but across the entire enterprise and beyond, dealing with issues such as interoperability and integration, governance and policy compliance.

Modelling is considered central in enterprise architecture, especially at the higher levels of abstraction for sharing systems-related knowledge across the enterprise (e.g., Zachman [125], ToGAF [86]). Most of the modeling relies on existing modeling methods, with static and dynamic ontologies. Some frameworks do emphasize the need for the modeling of "motivation" (column 6 in Zachman [125]). The Business Motivation Model, a recent OMG standard [83], has goals and means-ends relationships, but does not deal with social relationships. The use of *i** as intentional social models for enterprise architecture is suggested in [124]. Policy compliance using *i** based concepts are proposed in [28] [95].

Business model innovations and strategic change. Many business and industry sectors have been going through fundamental changes triggered by the Internet and now mobile technologies. eBay, Amazon, and Dell has been leading examples. The newspaper and publishing industries are seeing more dramatic transformations.

Conceptual modeling techniques have been used to describe and analyze business models [50], typically by introducing business specific ontologies, including such concepts as asset, revenue and value flow, channels, etc. Some prominent business authors have promoted graphical depictions of business goals [52].

Social modeling can complement these models by supporting analysis of strategic dependencies and analyzing alternative configurations that contribute differently to strategic business goals. The complementary use of *i** and the e^3value business modeling notation is outlined in [35]. An analysis of disruptive strategic change appears in [100]. Business model analysis leading to service-oriented system design was described in [66]. Business strategies of a networked value constellation were modelled using e^3value and a simplified version of *i** in [34].

6 Social Modeling for Security, Privacy, and Trust

Computer security has long been an active area of research. Many security models have been proposed. However, few have adopted a social perspective.

Security and privacy are ultimately human concerns. Despite advances in security and privacy technologies, breaches and failures continue to occur. Perfect security and privacy are acknowledged to be unattainable in practice [101]. Determined attackers have been able to overcome or bypass the strongest defensive mechanisms. Often, users themselves neglect or defeat the defensive measures when they interfere with work routines, or are too hard to use.

Social models allow the human issues of security, privacy, and trust to be systematically analyzed and addressed within an engineering process. In *i**, security, privacy, and trust can be modelled initially as high-level softgoals of some actors. Efforts to achieve them can be modelled in terms of refinement to more specific goals, such as confidentiality, integrity, availability, unlinkability, and so forth, eventually operationalizing them through implementable mechanisms such as encryption, firewalls, intrusion detectors, and anonymizers. The goals are accomplished via a network of hardware and software as well as human roles (security officers, network administrators, peer users, etc.) The dependencies among actors in such networks can be analyzed for viability, such as the adequacy or lack of reciprocal dependencies.

A social approach would recognize that security and privacy concerns are not necessarily high on every actor's agenda. They can be superseded by competing goals such as cost, task urgency, or convenience. An actor-based social model can reveal the trade-offs faced by each actor, this prompting system designers to seek solution alternatives that respond to the actor's overall needs and desires, not just those pertaining to security and privacy.

*i** modeling has been used to analyze and guide system design for security, privacy, and trust [118][117][21][93][99]. Goals and strategies of attackers (including insiders) can also be modelled and analyzed, to be taken into account during requirements analysis and design [65][21].

The Secure Tropos [77] approach added security specific constructs and introduced social ontology to security patterns [78]. Another line of work (also called Secure Tropos) provided extensions by defining ownership, permission, and delegation [30]. *i** has also been used as a starting point for deriving access controls [14]. Social modeling based on *i** were also applied to risk modeling and analysis [74].

In the TCD framework [26], *i** is used to model trust in a social network, with extensions to support quantitative simulation on actor behaviour, and changes over time in trust, distrust, and confidence in the network. A trust management framework which extends *i** by distinguishing delegation of permission from delegation of execution is described in [31]. [106] presents a cognitive model of trust expressed in an adapted *i** notation.

Intentional modeling ontologies, particularly goal models, have been developed for security requirements engineering [11][109]. The goal structures in these frameworks represent a single consolidated viewpoint, rather than distributed among multiple autonomous actors as in social models. Social modeling extends goal-based techniques by treating actors (such as users and attackers) as being autonomous but interdependent. Instead of finding best solutions in a graph structure from a single viewpoint, each actor seeks reconfigurations of social relationships that advance its strategic interests.

7 Research Issues

Social modeling, particularly in the form of *i** and variations, has been explored to some degree in research communities, mostly in the requirements engineering area. The preceding sections have highlighted selected work that use *i** or draw upon its basic concepts. Many have extended or adapted the basic *i** framework [122]. Industry adoption of social modeling remains limited. Most industry projects reporting experiences using *i** or related social modeling had close collaborations with academic researchers. Much remains to be done to make social modeling as widespread as static and dynamic modeling.

The *i** framework represents only one possible perspective on and approach for social modeling. It is hoped that many more new frameworks will emerge to allow a wide selection of modeling and analysis techniques, perhaps reflecting quite different underlying premises than those presented in section 2. In the following, a sampling of research issues arising from the *i** experience will be discussed. Many of these may be applicable to social modeling in general.

7.1 Usage Contexts and Methodologies

Formality. Conceptual models are abstractions which filter an understanding of the world through the lens of a small number of predefined concepts. Formal definition of the concepts, for example through an axiomatization in a formal language and logic will facilitate automated inference and tool support. A high degree of formality, however, requires specialized training and thus restricts the user population. *i** variants have ranged over a broad spectrum of formal to informal approaches.

Most widely used conceptual modeling notations – from Data Flow Diagrams to UML, are semi-formal, and rely heavily on graphical visual notations. Formality is more difficult to attain in social modeling, as there is little agreement on any precise characterization of social reality, or even its possibility and desirability.

How much formality, of what kind, for use in what context – these are crucial research questions to pursue in order to create practical social modeling methodologies. A simplified, fairly informal notation may be necessary to encourage untrained stakeholder participation and interaction, while a more formal version may be needed for greater expressive power, better tool support, larger scale projects, and more automated analysis. A similar approach is taken in BPMN [5].

Domain terms. Aside from the predefined modeling constructs, linguistic terms chosen by modellers to represent domain concepts can also present difficulties. To reflect stakeholder perspective and promote active participation and ownership of the models, faithfulness to the language used by stakeholder is important. On the other hand, to facilitate analysis and to share knowledge across projects, the analyst may need to rephrase the domain terms. In any case, where most visual presentations of conceptual models require concise phrases to embody a concept, the choice of an appropriate phrase that will accurately convey the intended meaning can be quite demanding. The adoption of a project lexicon or ontology [6] is worth considering. Methodologically, one may want to have different sets of models using different domain vocabularies, e.g., one set for stakeholder participation, another for sharing and reuse. Coordination among sets of models will be another research issue. Lightweight natural language processing may also be helpful [102].

Patterns. Creating models from scratch can be quite labour intensive. A common solution is to build up collections of reusable models or generic patterns. The pattern approach for *i** has been explored in a number of works [89][73][78][58]. Patterns represent generalized knowledge, so they must be re-contextualized when applied to a specific situation. There are questions of validity of a pattern, and of applicability to a specific circumstance. There is risk that reliance on available patterns may distort analysis of the unique circumstances of a specific situation [104]. *i** has also been used to formalize the representation of problem contexts, forces, and alternate solutions in design patterns [80].

Visual Presentation and Interaction. Graphical models rely on effective human interpretation, interactive manipulation, and visual analysis. Their visual and cognitive properties are emerging research topics [76].

One ongoing challenge in *i** modeling is model scalability. *i** models are inherently networks, reflecting its conception of multilateral social relationships. Strategic rationale models may have dominating tree structures, but softgoals can receive contributions from all levels in the decomposition hierarchy, resulting in general graph structures. It is therefore difficult to take advantage of hierarchical abstraction mechanisms that provide much of the structural simplification in traditional structured analysis techniques (e.g., [17]). These challenges can potentially be overcome by inventive use of view mechanisms [113][61][13] or aspect orientation [1].

7.2 Conceptual Limitations and Extensions

The reduction of a complex world into a small number of modeling concepts is necessarily a compromise – one is faced with tough choices on what to include or exclude. The original *i** framework reflected principles described in Section 2. In practice, some users have found *i** too simple and limited in expressiveness, requiring extensions to make further conceptual distinctions, especially in specialized areas such as security [30][21]. Others found it too complex, electing to use subsets of the *i** constructs, e.g., [19][34][43]. By comparison, DFDs and ER models have 3 to 4 main conceptual constructs, whereas UML and BPMN have many more. In this section, we consider some areas for further exploration.

Reasoning. Although the process of constructing a model can in itself contribute significantly to understanding the issues in a domain [20], a deeper understanding can be gained by analyzing the reasoning implied by the intentional structure of the model. The SR model in *i** is an explicit representation of means-ends reasoning and contributions to quality goals, albeit inherently partial and incomplete. The SD model provides pathways for propagating intentionality across actors.

A number of approaches have been developed for reasoning over goal models. The NFR framework [11] offers an interactive procedure for propagating labels across the NFR goal graph. Based on the link types and labels, propagation steps can be automated if they do not require human judgment, though they can be overridden manually if desired. Fully automated procedures have been developed [32], some using assigned weights and numerical values [97].

A combined use of interactive and automated methods is likely desirable. Interactive method with a high degree of human judgement may be best suited to early RE due to its participatory, informal nature, when the model is very incomplete and in the process of being iteratively elaborated [42]. A highly interactive procedure will engage the modeller more fully and contributes to understanding at every step. When the models are more stable in later stages, automated evaluation of goal satisfaction can greatly improve efficiency of the process. The semantics of goal models is an active research topic. More empirical studies are also warranted.

Beliefs, Assumptions, Justifications. While intentional ontologies have emphasized goals and goal-based reasoning, beliefs have not been so well investigated. Some goal modeling frameworks have given prominence to assumptions, justifications, and context (e.g., [54]). Beliefs appear in the NFR framework in the form of claims, and are subject to the same evaluation propagation procedure as softgoals. Further exploration of the semantics and implications for practical reasoning and analysis are needed in the context of social modeling.

Viewpoints. As outlined in the premises, actor autonomy implies that each actor is reasoning from its own perspective. Therefore the rationales of each actor are modeled separately, within its own boundary scope in the *i** SR model. This is in contrast to goal-oriented RE frameworks (such as KAOS or the NFR framework) which employ ontologies which are intentional but not social.

In the current formulation of the *i** framework, this premise is only partially supported, as the model admits only one perspective on each actor's reasoning. This is an oversimplification, given the premise that each actor has limited access to other actor's internal rationales. To fully adhere to the premise, each actor would have its own model of every other actor's rationales, i.e., we would need as many SR models as there are actors, each from one actor's viewpoint. The SR models could be the result of modeling from interview data obtained from each actor separately. A more elaborate methodology would provide guidelines and support for merging models, and how to manage multiple viewpoints to benefit from inconsistencies and disagreements in the process [98].

A related topic is the modeling of cross-cutting concerns. Aspect-oriented techniques have been used to extend the expressiveness of *i** [79] and to simplify *i** models [1].

Process dynamics. One fundamental question in the design of a modeling language is the extent to which various basic ontologies should be incorporated and how tightly they should be integrated. Under Structured Analysis, DFDs and ERDs addressed dynamic and static ontologies quite separately. A tighter integration of the two basic ontologies was one of the objectives of object-oriented frameworks. In *i**, the social ontology is closely tied to an intentional ontology of goals and rationales, but dynamic and static ontologies are not explicitly incorporated.

Lack of temporal concepts is often felt to be an inhibiting factor in understanding *i** models. When *i** is used to model a business process, only the social and intentional relationships are portrayed. There is no indication of the temporal progression of the process, no beginning or end. Separate models are needed to express the static relationships and dynamics.

Concerns with incorporating features from other ontologies into a social modeling framework include increased complexity, commitment to a single version of the other ontologies, and not being able to do justice to the other ontologies with a limited set of features. One approach is to provide a loose coupling with an existing language or notation that offers rich features based on other ontologies. The User Requirements Notation (URN) brings together the social and intentional modeling of *i** (in the form of GRL) and the scenario-oriented dynamic ontology of Use Case Maps (UCM). Mappings and linkages between the two ontologies are provided through a unified metamodel [46][63]. However, small extensions to a social ontology for a specific purpose can be effective. For example, [27] and [110] represent two approaches on temporal extensions to *i**, allowing process simulation. The former adds a *precedes* operator, while the latter approach adopts a fuller set of procedural operators from the ConGolog language (sequence, non-deterministic pick, test, repeat, etc.).

Evolution and Change. Most applications of social modeling are concerned about change – how to improve the social configuration to the benefit of stakeholders. The representation of change in *i** is limited. Alternate social configurations (e.g., "as-is", "to-be", "could-be", etc.) are typically depicted in separate SD models. An SR model can show multiple alternatives and how they contribute differently to various stakeholders' strategic interests, though the representation is limited by visual complexity.

The change from As-Is to To-Be is an abrupt structural change, with no representation or reasoning about the steps that may be needed to bring about the change. Changes in the environment, especially gradual continuous change, are hard to represent. Complementary use of system dynamics and social intentional modeling is a topic to be explored. A method for using *i** in adaptive system modeling has been proposed [33]. Modeling strategic change is studied in [100].

Roles. In complex social settings, a role is distinguished from the person who plays the role. In an organization, position occupied by a person typically covers multiples roles. For example, a project manager runs a project, but may or may not be the performance evaluator for employees. The distinctions are useful for separating intentional dependencies on a role from those on the agent that plays the role, and for identifying role conflicts. An organization can be modeled as an aggregate position, i.e., a set of positions related via dependencies, regardless of which individual persons are occupying those positions. Roles are also useful for analyzing security, emphasizing a social analysis perspective [64][65][31].

The *i** framework offers notations for distinguishing roles, agents, and positions, and the association links between them (plays, occupies, covers), but the meanings of these concepts are not well defined and remain open for interpretation [120].

Inheritance. Inheritance along a specialization dimension is an important mechanism in conceptual models and is heavily used in static and dynamic ontologies, especially in object-oriented modeling. Due to the premise of actor autonomy, the usual inheritance concept does not apply straightforwardly to intentional relationships among actors [64]. Some research issues are identified in [68]. In general, abstraction mechanisms (such as classification/instantiation, generalization/specialization, part-whole, etc.) that are well studied in conceptual modeling [81] for static and dynamic ontologies are not yet well developed for social modeling.

7.3 Model Management and Tools

Conceptual models serve multiple purposes. They may be used to facilitate communication – between analysts and stakeholders, among analysts, developers and project managements, within a project or across projects within an enterprise. They may be used to describe and understand existing situations, to uncover problems and issues, or to explore hypothetical scenarios and potential solutions. Some models are used in impromptu settings, such as sketches on a whiteboard. Others are meant to be records in a repository for later retrieval or reuse.

Social models produced for different purposes may well need different kinds of tool support and management methods. For many small scale applications of *i**, general purpose drawing tools such as Visio have been found to be adequate and flexible, with the advantage of broad availability, and not requiring special installation and learning.

About a dozen software tools have been developed to support *i** modeling and specialized functionalities. Several are open source, some based on the Eclipse platform (e.g., OpenOME [87], TAOM4E [107], jUCMNav [51], jPRIM [47]). Some have built on the programmability of general purpose tools, e.g., [67][94]. *i**-related tools and approaches are compared at the *i** wiki [44] and in [36][103].

With many extensions and variations, the diversity of metamodels for *i**-related notations and tools has arisen as a challenge. Proposals have been made to reconcile differences [69] and to have a common interchange format [10]. Integration with other modeling frameworks using a metamodeling approach using Telos [82] have been investigated [91][55]. When a model progresses through a series of versions, version management issues arise. Merging different versions of a model has been investigated [98]. View mechanisms have been studied in [113][61][13]. A commercial requirements management system (Telelogic DOORS) has been used as a repository to manage change across multiple modeling frameworks in [97] and [28].

8 Conclusions

As computing and information systems interact more intricately with the social world, social modeling has arisen as a new area for conceptual modeling. Experiences with the *i** framework have revealed encouraging possibilities as well as many research challenges.

Conceptual modeling is of course only one way to bring understanding of complex social phenomena into the system design process. Techniques such as participatory design, ethnography, and others can equally enrich the process of system design. Conceptual modeling approaches have the potential of a more direct integration into established system engineering methods, supporting fine-grained analysis and traceability. As social modeling evolves, much can be gained by further exploring the synergies between conceptual modeling methods and the rich understanding of the human social experience from the social sciences and humanities.

Acknowledgements. My sincere thanks goes to everyone who has contributed to the development of *i**, directly or indirectly, by using, adapting, extending, and debating it, and uncovering its limitations and shortcomings. Most of all, I am deeply indebted to Professor John Mylopoulos, whose vision and open-mindedness made possible my forays into social modeling. Funding sources for the research include the Natural Sciences and Engineering Research Council (NSERC) of Canada and Bell University Laboratories. I am also grateful to Neil Maiden, Daniel Amyot, and Lin Liu, who kindly provided feedback on a draft of this paper on very short notice. Figures 1 and 2 are taken from [114], with kind permission of Springer Science+Business Media.

References

1. Alencar, F., Castro, J., Moreira, A., Araújo, J., Silva, C., Ramos, R., Mylopoulos, J.: Integration of Aspects with i* Models. In: Kolp, M., Henderson-Sellers, B., Mouratidis, H., Garcia, A., Ghose, A.K., Bresciani, P. (eds.) AOIS 2006. LNCS, vol. 4898, pp. 183–201. Springer, Heidelberg (2008)
2. Amyot, D.: Introduction to the User Requirements Notation: Learning by Example. Computer Networks 42(3), 285–301 (2003)
3. An, Y., Dalrymple, P.W., Rogers, M., Gerrity, P., Horkoff, J., Yu, E.: Collaborative Social Modeling for Designing a Patient Wellness Tracking System in a Nurse-Managed Health Care Center. In: 4th Int. Conf. on Design Science Research in Information Systems and Technology (DESRIST) (2009)

4. Beck, K., Boehm, B.: Agility Through Discipline. IEEE Computer 44–46 (June 2003)
5. BPMN: Business Process Modeling Notation specification (2009), `http://www.bpmn.org`
6. Breitman, K., Leite, J.C.S.P.: Ontology as a Requirements Engineering Product. In: IEEE Int. Conf. Requirements Eng. RE 2003, pp. 309–319 (2003)
7. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: TROPOS: an agent-oriented software development methodology. J. Autonomous Agents and Multiagent Systems 8(3), 203–236 (2004)
8. Briand, L.C., Yong-Mi Kim, Y.M., Melo, W.L., Seaman, C.B., Basili, V.R.: Q-MOPP: qualitative evaluation of maintenance organizations, processes and products. Journal of Software Maintenance 10(4), 249–278 (1998)
9. Bubenko, J.A.: Information Modeling in the Context of System Development. IFIP Congress, 395–411 (1980)
10. Cares, C., Franch, X., Perini, A., Susi, A.: iStarML: An XML-based Model Interchange Format for i*. In: Castro, J.B., Franch, X., Perini, A., Yu, E. (eds.) Proc. 3rd Int. i* Workshop, Recife, Brazil, February 11-12, 2008, vol. 322, pp. 13–16. CEUR Workshop Proceedings, CEUR-WS.org (2008)
11. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Dordrecht (1999)
12. Coad, P., Yourdon, E.: Object-Oriented Analysis, 2nd edn. Prentice-Hall, Englewood Cliffs (1991)
13. Cocca, C.: Towards Improved Visual Support for i* Modeling. MISt thesis. Faculty of Information, University of Toronto (2007)
14. Crook, R., Ince, D., Nuseibeh, B.: On modelling access policies: Relating roles to the organisational context. In: IEEE Int. Requirements Eng. Conf. RE 2005, pp. 157–166 (2005)
15. Curtis, W., Kellner, M.I., Over, J.: Process Modeling. Commun. ACM 35(9), 75–90 (1992)
16. Cysneiros, L.M., Yu, E.: Addressing Agent Autonomy in Business Process Management - with Case Studies on the Patient Discharge Process. In: Proc. of Information Resources Management Association Conference, New Orleans, pp. 436–439 (2004)
17. Demarco, T.: Structured Analysis and System Specification. Prentice-Hall, Englewood Cliffs (1979)
18. DesCARTES Architect. Catholic University of Louvain, Belgium, `http://www.isys.ucl.ac.be/descartes/`
19. Donzelli, P.: A goal-driven and agent-based requirements engineering framework. Requirements Engineering 9(1), 16–39 (2004)
20. Easterbrook, S.M., Yu, E., Aranda, J., Fan, Y., Horkoff, J., Leica, M., Qadir, R.A.: Do Viewpoints Lead to Better Conceptual Models? An Exploratory Case Study. In: IEEE Int. Requirements Eng. Conf., pp. 199–208 (2005)
21. Elahi, G., Yu, E.: A Goal Oriented Approach for Modeling and Analyzing Security Trade-Offs. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 375–390. Springer, Heidelberg (2007)
22. Elahi, G., Yu, E., Annosi, M.C.: Modeling Knowledge Transfer in a Software Maintenance Organization - An Experience Report and Critical Analysis. In: Stirna, J., Persson, A. (eds.) PoEM 2008. LNBIP, vol. 15, pp. 15–29. Springer, Heidelberg (2008)
23. Estrada, H., Martínez, A., Pastor, O., Mylopoulos, J.: An Empirical Evaluation of the i* Framework in a Model-based Software Generation Environment. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 513–527. Springer, Heidelberg (2006)
24. Franch, X.: On the Lightweight Use of Goal-Oriented Models for Software Package Selection. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 551–566. Springer, Heidelberg (2005)

25. Fuxman, A., Liu, L., Mylopoulos, J., Pistore, M., Roveri, M., Traverso, P.: Specifying and analyzing early requirements in Tropos. Requirements Engineering Journal 9(2), 132–150 (2004)
26. Gans, G., Jarke, M., Kethers, S., Lakemeyer, G.: Continuous requirements management for organisation networks: a (dis)trust-based approach. Requirements Engineering Journal 8(1), 4–22 (2003)
27. Gans, G., Jarke, M., Lakemeyer, G., Schmitz, D.: Deliberation in a metadata-based modeling and simulation environment for inter-organizational networks. Inf. Syst. 30(7), 587–607 (2005)
28. Ghanavati, S., Amyot, D., Peyton, L.: Towards a Framework for Tracking Legal Compliance in Healthcare. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 218–232. Springer, Heidelberg (2007)
29. Ghose, A., Koliadis, G.: Actor Eco-systems: From High-Level Agent Models to Executable Processes via Semantic Annotations. IEEE COMPSAC (2), 177–184 (2007)
30. Giorgini, P., Massacci, F., Mylopoulos, J., Zannone, N.: Modeling Security Requirements Through Ownership, Permission and Delegation. In: IEEE Int. Requirements Eng. Conf. RE 2005, France, pp. 167–176 (2005)
31. Giorgini, P., Massacci, F., Mylopoulos, J., Zannone, N.: Requirements engineering for trust management: model, methodology, and reasoning. Int. J. of Information Security 5(4), 25, 274 (2006)
32. Giorgini, P., Nicchiarelli, E., Mylopoulos, J., Sebastiani, R.: Formal Reasoning Techniques for Goal Models. Journal of Data Semantics 1, 1–20 (2003)
33. Goldsby, H.J., Sawyer, P., Bencomo, N., Cheng, B.H.C., Hughes, D.: Goal-based Modeling of Dynamically Adaptive System Requirements. In: 15th IEEE Int. Conf. on Engineering of Computer Based Systems, pp. 36–45 (2008)
34. Gordijn, J., Petit, M., Wieringa, R.: Understanding business strategies of networked value constellations using goal- and value modeling. In: IEEE Int. Conf. on Requirements Eng. RE 2006, pp. 126–135 (2006)
35. Gordijn, J., Yu, E., Van Der Raadt, B.: E-service design using i* and e3value modeling. IEEE Software 23(3), 26–33 (2006)
36. Grau, G., Cares, C., Franch, X., Navarrete, F.J.: A Comparative Analysis of i* Agent-Oriented Modelling Techniques. In: Int. Conf. on Software Eng. and Knowledge Eng., San Francisco Bay, California, USA, pp. 657–663 (2006)
37. Grau, G., Franch, X., Maiden, N.A.M.: PRiM: An i*-based process reengineering method for information systems specification. Inf. & Softw. Tech. 50(1-2), 76–100 (2008)
38. Greenspan, S.J., Mylopoulos, J., Borgida, A.: Capturing More World Knowledge in the Requirements Specification. In: ACM/IEEE Int. Conf. Softw. Eng., pp. 225–235 (1982)
39. Guizzardi, R.S.S.: Agent-oriented Constructivist Knowledge Management. Ph.D. thesis, Enschede: University of Twente. The Netherlands (2006)
40. Hammer, M.: Reengineering work: Don't Automate, Obliterate. Harvard Business Review, pp. 104–112 (July 1990)
41. Henderson-Sellers, B., Giorgini, P. (eds.): Agent-Oriented Methodologies. Idea Group Inc., Hershey (2005)
42. Horkoff, J.: Using i* Models for Evaluation. M.Sc. Thesis, Dept. of Computer Science, University of Toronto (2006)
43. Horkoff, J., Elahi, G., Abdulhadi, S., Yu, E.: Reflective Analysis of the Syntax and Semantics of the i* Framework. In: Song, I.-Y., Piattini, M., Chen, Y.-P.P., Hartmann, S., Grandi, F., Trujillo, J., Opdahl, A.L., Ferri, F., Grifoni, P., Caschera, M.C., Rolland, C., Woo, C., Salinesi, C., Zimányi, E., Claramunt, C., Frasincar, F., Houben, G.-J., Thiran, P. (eds.) ER Workshops 2008. LNCS, vol. 5232, pp. 249–260. Springer, Heidelberg (2008)
44. i* wiki, `http://istar.rwth-aachen.de`
45. IEEE: Guide to Software Requirements Specifications. IEEE Standard 830-1993. In: Software Engineering Standards. IEEE Computer Society Press, Los Alamitos (1993)

46. International Telecommunications Union (ITU-T) Recommendation Z.151: User Requirements Notation (URN) - Language Definition (2008)
47. J-PRiM. A Process Reengineerng i* Modeling Tool, `http://www.ideaciona.com/PhD/JPRIM/`
48. Jackson, M.: System Development. Prentice-Hall, Englewood Cliffs (1983)
49. Jiang, L., Topaloglou, T., Borgida, A., Mylopoulos, J.: Goal-Oriented Conceptual Database Design. In: IEEE Int. Conf. on Requirements Eng., pp. 195-204 (2007)
50. Johannesson, P.: The Role of Business Models in Enterprise Modelling. In: Krogstie, J., et al. (eds.) Conceptual Modelling in Info. Systems Eng., pp. 123–140. Springer, Heidelberg (2007)
51. jUCMNav. University of Ottawa, `http://jucmnav.softwareengineering.ca/jucmnav/`
52. Kaplan, R.S., Norton, D.P.: Having trouble with your strategy? Then map it. Harvard Business Review, 167–176 (September-October 2002)
53. Kazhamiakin, R., Pistore, M., Roveri, M.: A Framework for Integrating Business Processes and Business Requirements. In: IEEE Int. Enterprise Distributed Object Computing Conf., pp. 9–20 (2004)
54. Kelly, T.P., McDermid, J.A.: A Systematic Approach to Safety Case Maintenance. In: Felici, M., Kanoun, K., Pasquini, A. (eds.) SAFECOMP 1999. LNCS, vol. 1698, pp. 13–26. Springer, Heidelberg (1999)
55. Kethers, S.: Multi-perspective modelling and analysis of cooperation processes. Doctoral dissertation, RWTH Aachen University, Germany (2000)
56. Kethers, S., Gans, G., Schmitz, D., Sier, D.: Modelling trust relationships in a healthcare network: Experiences with the TCD framework. In: Bartmann, D., et al. (eds.) European Conf. on Information Systems (ECIS), Regensburg, Germany, pp. 1321–1328 (2005)
57. Kling, R. (ed.): Computerization and Controversy: Value Conflicts and Social Choices, 2nd edn. Morgan Kaufmann, San Francisco (1996)
58. Kolp, M., Giorgini, P., Mylopoulos, J.: Organizational Patterns for Early Requirements Analysis. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 617–632. Springer, Heidelberg (2003)
59. Koubarakis, M., Plexousakis, D.: A formal framework for business process modelling and design. Information Systems 27(5), 299–319 (2002)
60. Lapouchnian, A., Yu, Y., Mylopoulos, J.: Requirements-Driven Design and Configuration Management of Business Processes. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 246–261. Springer, Heidelberg (2007)
61. Leica, M.F.: Scalability concepts for i* modeling and analysis. M.Sc. thesis. Dept. of Computer Science, University of Toronto (2005)
62. Lespérance, Y., Kelley, T., Mylopoulos, J., Yu, E.: Modeling dynamic domains with ConGolog. In: Jarke, M., Oberweis, A. (eds.) CAiSE 1999. LNCS, vol. 1626, pp. 365–380. Springer, Heidelberg (1999)
63. Liu, L., Yu, E.: Designing Information Systems in Social Context: A Goal and Scenario Modelling Approach. Information Systems 29(2), 187–203 (2004)
64. Liu, L., Yu, E., Mylopoulos, J.: Analyzing security requirements as relationships among strategic actors. In: Proc. 2nd symposium on requirements engineering for information security (SREIS 2002), Raleigh, North Carolina (2002)
65. Liu, L., Yu, E., Mylopoulos, J.: Security and privacy requirements analysis within a social setting. In: IEEE Int. Conf. on Requirements Eng. RE 2003, pp. 151–161 (2003)
66. Lo, A., Yu, E.: From Business Models to Service-Oriented Design: A Reference Catalog Approach. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 87–101. Springer, Heidelberg (2007)
67. Lockerbie, J.A., Maiden, N.A.M.: REDEPEND: Extending i* Modelling into Requirements Processes. In: IEEE Int. Conf. on Requirements Eng., pp. 361–362 (2006)

68. López, L., Franch, X., Marco, J.: Defining Inheritance in i* at the Level of SR Intentional Elements. In: Castro, J.B., Franch, X., Perini, A., Yu, E. (eds.) Proc. 3rd Int. i* Workshop, Recife, Brazil. CEUR Workshop Proceedings, vol. 322, pp. 71–74. CEUR-WS.org (2008)
69. Lucena, M., Santos, E., Silva, C., Alencar, F., Silva, M.J., Castro, J.: Towards a unified metamodel for i*. In: IEEE Int. Conf. On Research Challenges in Information Science, RCIS 2008, pp. 237–246 (2008)
70. Lyytinen, K.: Different Perspectives on Information Systems: Problems and Solutions. ACM Computing Surveys 19(1), 5–46 (1987)
71. Maiden, N.A.M., Jones, S., Manning, S., Greenwood, J., Renou, L.: Model-Driven Requirements Engineering: Synchronising Models in an Air Traffic Management Case Study. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 368–383. Springer, Heidelberg (2004)
72. Maiden, N.A.M., Jones, S.: The RESCUE Requirements Engineering Process: An Integrated User-Centred Requirements Engineering Process for Eurocontrol, Version 4.1 (2004), http://hcid.soi.city.ac.uk/research/Rescue.html
73. Maiden, N., Manning, S., Jones, S., Greenwood, J.: Generating Requirements from Systems Models Using Patterns: A Case Study. Requirements Eng. Journal 10(4), 276–288 (2005)
74. Matulevicius, R., Mayer, N., Mouratidis, H., Dubois, E., Heymans, P., Genon, N.: Adapting Secure Tropos for Security Risk Management during Early Phases of the Information Systems Development. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 541–555. Springer, Heidelberg (2008)
75. Medina-Mora, R., Winograd, T., Flores, R., Flores, F.: The action workflow approach to workflow management technology. In: ACM Conf. on Computer-Supported Cooperative Work, Toronto, Canada, pp. 281–288 (1992)
76. Moody, D.L.: Cognitive Load Effects on End User Understanding of Conceptual Models: An Experimental Analysis. In: Benczúr, A.A., Demetrovics, J., Gottlob, G. (eds.) ADBIS 2004. LNCS, vol. 3255, pp. 129–143. Springer, Heidelberg (2004)
77. Mouratidis, H., Giorgini, P., Manson, G.: When security meets software engineering: A case of modelling secure information systems. Information Systems 30(8), 609–629 (2007)
78. Mouratidis, H., Weiss, M., Giorgini, P.: Security patterns meet agent oriented software engineering: A complementary solution for developing security information systems. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 225–240. Springer, Heidelberg (2005)
79. Mussbacher, G.: Aspect-Oriented User Requirements Notation: Aspects in Goal and Scenario Models. In: Giese, H. (ed.) MODELS 2008. LNCS, vol. 5002, pp. 305–316. Springer, Heidelberg (2008)
80. Mussbacher, G., Amyot, D., Weiss, M.: Formalizing Patterns with the User Requirements Notation. In: Taibi, T. (ed.) Design Pattern Formalization Techniques, pp. 304–325. IGI Publishing (2007)
81. Mylopoulos, J.: Information Modeling in the Time of the Revolution. Inf. Syst. 23(3-4), 127–155 (1998)
82. Mylopoulos, J., Borgida, A., Jarke, M., Koubarakis, M.: Telos: Representing Knowledge about Information Systems. ACM Trans. on Information Systems 8(4), 325–362 (1990)
83. OMG. Business Motivation Model (BMM) (2006), http://www.omg.org/spec/BMM/
84. Object Management Group (OMG): SPEM: Software Process Engineering Metamodel, Version 2.0 (2008)
85. Object Management Group (OMG), Unified Modeling Language, http://www.uml.org
86. Open Group. The Open Group Architecture Framework. version 9 (2009), http://www.opengroup.org
87. OpenOME. University of Toronto, http://www.cs.toronto.edu/km/openome/

88. Pardillo, J., Trujillo, J.: Integrated Model-Driven Development of Goal-Oriented Data Warehouses and Data Marts. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 426–439. Springer, Heidelberg (2008)
89. Pavan, P., Maiden, N.A.M., Zhu, X.: Towards a Systems Engineering Pattern Language: Applying i* to Model Requirements Architecture Patterns. In: ICSE STRAW 2003: 2nd Int. Ws. From Software Requirements to Architectures, Portland, Oregon, USA, pp. 134–141 (2003)
90. Perini, A., Susi, A.: Developing a decision support system for integrated production in agriculture. Environmental Modelling and Software 19(9), 821–829 (2004)
91. Petit, M.: Formal Requirements Engineering of Manufacturing Systems: A Multi-Formalism and Component-Based Approach, PhD dissertation. University of Namur, Belgium (1999)
92. Pourshahid, A., Chen, P., Amyot, D., Forster, A.J., Ghanavati, S., Peyton, L., Weiss, M.: Toward an integrated User Requirements Notation framework and tool for Business Process Management. In: 3rd Int. MCeTech Conf. on eTechnologies, Montréal, Canada, pp. 3–15. IEEE Computer Society, Los Alamitos (2008)
93. Pourshahid, A., Tran, T.: Modeling Trust in E-Commerce: An Approach Based on User Requirements. In: 9th ACM Int. Conf. on Electronic Commerce (ICEC 2007), pp. 413–421 (2007)
94. REDEPEND-REACT. An Architecture Analysis Tool, `http://www.ideaciona.com/PhD/REDEPEND-REACT/`
95. Rifaut, R., Dubois, E.: Using Goal-Oriented Requirements Engineering for Improving the Quality of ISO/IEC 15504 based Compliance Assessment Frameworks. In: IEEE Int. Conf. on Requirements Eng. RE 2008, pp. 33–42 (2008)
96. Rolland, C.: Capturing System Intentionality with Maps. In: Krogstie, J., Opdahl, A.L., Brinkkemper, S. (eds.) Conceptual Modelling in Information Systems Engineering, pp. 141–158. Springer, Heidelberg (2007)
97. Roy, J.-F., Kealey, J., Amyot, D.: Towards Integrated Tool Support for the User Requirements Notation. In: Gotzhein, R., Reed, R. (eds.) SAM 2006. LNCS, vol. 4320, pp. 198–215. Springer, Heidelberg (2006)
98. Sabetzadeh, M., Easterbrook, S.: View merging in the presence of incompleteness and inconsistency. Requirements Engineering 11(3), 174–193 (2006)
99. Samavi, R., Topaloglou, T.: Designing Privacy-Aware Personal Health Record Systems. In: Song, I.-Y., Piattini, M., Chen, Y.-P.P., Hartmann, S., Grandi, F., Trujillo, J., Opdahl, A.L., Ferri, F., Grifoni, P., Caschera, M.C., Rolland, C., Woo, C., Salinesi, C., Zimányi, E., Claramunt, C., Frasincar, F., Houben, G.-J., Thiran, P. (eds.) ER Workshops 2008. LNCS, vol. 5232, pp. 12–21. Springer, Heidelberg (2008)
100. Samavi, R., Yu, E., Topaloglou, T.: Strategic Reasoning about Business Models: A Conceptual Modeling Approach. Information Systems and e-Business Management 7(2), 171–198 (2009)
101. Sandhu, R.S.: Good-Enough Security: Toward a Pragmatic Business-Driven Discipline. IEEE Internet Computing 7(1), 66–68 (2003)
102. Sawyer, P., Rayson, P., Cosh, K.: Shallow knowledge as an aid to deep understanding in early phase requirements engineering. IEEE Trans. on Softw. Eng. 31(11), 969–981 (2005)
103. Schmitz, D., Lakemeyer, G., Jarke, M.: Comparing TCD/SNet with two other formal analysis approaches based on i*: Formal Tropos and Secure Tropos. In: Latour, T., Petit, M. (eds.) 8th Workshop on Agent-Oriented Information Systems (AOIS@CAiSE), pp. 29–40. Presses Universitaires de Namur (2006)
104. Strohmaier, M., Horkoff, J., Yu, E., Aranda, J., Easterbrook, S.M.: Can Patterns Improve i* Modeling? Two Exploratory Studies. In: Paech, B., Rolland, C. (eds.) REFSQ 2008. LNCS, vol. 5025, pp. 153–167. Springer, Heidelberg (2008)

105. Strohmaier, M., Yu, E.S., Horkoff, J., Aranda, J., Easterbrook, S.M.: Analyzing Knowledge Transfer Effectiveness: An Agent-Oriented Modeling Approach. In: 40th Hawaii Int. Conf. on Sys. Sci. HICSS 2007, p. 188. IEEE Computer Society, Los Alamitos (2007)
106. Sutcliffe, A.G.: Trust: From Cognition to Conceptual Models and Design. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 3–17. Springer, Heidelberg (2006)
107. TOAM4E. Tool for Agent Oriented Modeling. FBK-IRST, Italy, `http://sra.itc.it/tools/taom4e/`
108. van Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Guided Tour. In: 5th IEEE Int. Symp. on Requirements Eng. RE 2001, Toronto, pp. 249–263 (2001)
109. van Lamsweerde, A.: Elaborating Security Requirements by Construction of Intentional Anti-Models. In: 26th Int. Conf. on Software Eng. ICSE 2004, pp. 148–157. ACM/ IEEE, Edinburgh (2004)
110. Wang, X., Lespérance, Y.: Agent-oriented requirements engineering using ConGolog and i*. In: Wagner, G., Karlapalem, K., Lespérance, Y., Yu, E. (eds.) Agent-Oriented Information Systems Workshop (AOIS 2001), Montreal, Canada, pp. 59–78. iCue Publishing, Berlin (2001)
111. Weiss, M., Amyot, D.: Business process modeling with URN. International Journal of E-Business Research 1(3), 63–90 (2005)
112. Wieringa, R.: Requirements Engineering: Problem Analysis and Solution Specification. In: Koch, N., Fraternali, P., Wirsing, M. (eds.) ICWE 2004. LNCS, vol. 3140, pp. 13–16. Springer, Heidelberg (2004)
113. You, J.Z.: Using meta-schema driven views for scaling i* models. M.Sc. thesis. Dept. of Computer Science, University of Toronto (2004)
114. Yu, E.: Agent-Oriented Modelling: Software Versus the World. In: Wooldridge, M.J., Weiß, G., Ciancarini, P. (eds.) AOSE 2001. LNCS, vol. 2222, pp. 206–225. Springer, Heidelberg (2002)
115. Yu, E.: Agent Orientation as a Modelling Paradigm. Wirtschaftsinformatik 43(2), 123–132 (2001)
116. Yu, E.S.K.: Models for Supporting the Redesign of Organizational Work. In: Conf. on Organizational Computing Systems (COOCS 1995), pp. 225–236. ACM Press, New York (1995)
117. Yu, E., Cysneiros, L.M.: Designing for Privacy in the Presence of Other Requirements. In: Falcone, R., Barber, S., Korba, L., Singh, M.P. (eds.) AAMAS 2002. LNCS, vol. 2631, pp. 209–223. Springer, Heidelberg (2003)
118. Yu, E., Liu, L.: Modelling Trust for System Design Using the *i** Strategic Actors Framework. In: Falcone, R., Singh, M., Tan, Y.-H. (eds.) Trust in Cyber-societies. LNCS, vol. 2246, pp. 175–194. Springer, Heidelberg (2001)
119. Yu, E.S.K., Mylopoulos, J.: From E-R to A-R: Modelling Strategic Actor Relationships for Business Process Reengineering. In: Loucopoulos, P. (ed.) ER 1994. LNCS, vol. 881, pp. 548–565. Springer, Heidelberg (1994)
120. Yu, E.S.K., Mylopoulos, J.: Understanding 'Why' in Software Process Modelling, Analysis, and Design. In: IEEE Int. Conf. Softw. Eng., pp. 159–168 (1994)
121. Yu, E.S.K., Mylopoulos, J.: Using Goals, Rules and Methods to Support Reasoning in Business Process Reengineering. Int. J. of Intelligent Systems in Accounting, Finance, and Management 5(1), 1–13 (1996)
122. Yu, E.S.K.: Modelling Strategic Relationships For Process Reengineering. Ph.D. dissertation. Dept. of Computer Science, University of Toronto (1995)
123. Yu, E.S.: Towards Modelling And Reasoning Support For Early-Phase Requirements Engineering. In: 3rd IEEE Int. Symp. on Requirements Eng., pp. 226–235 (1997)
124. Yu, E.S.K., Strohmaier, M., Deng, X.: Exploring Intentional Modeling and Analysis for Enterprise Architecture. In: Workshop on Trends in Enterprise Architecture Research (TEAR), 10th IEEE Int. Enterprise Distributed Object Computing Conference, October 2006, pp. 32.1– 32.8. IEEE Comp. Soc., Los Alamitos (2006)
125. Zachman, J.A.: A Framework for Information Systems Architecture. IBM Systems Journal 26(3) (1987)