Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering

Eric S. K. Yu
Faculty of Information Studies, University of Toronto
Toronto, Ontario, Canada M5S 3G6
eric.yu@utoronto.ca

Abstract

Requirements are usually understood as stating what *a system is supposed to do, as opposed to* how *it should do it. However, understanding the organizational context and rationales (the* "Whys"*) that lead up to systems requirements can be just as important for the ongoing success of the system. Requirements modelling techniques can be used to help deal with the knowledge and reasoning needed in this earlier phase of requirements engineering. However, most existing requirements techniques are intended more for the later phase of requirements engineering, which focuses on completeness, consistency, and automated verification of requirements. In contrast, the early phase aims to model and analyze stakeholder interests and how they might be addressed, or compromised, by various system-and-environment alternatives. This paper argues, therefore, that a different kind of modelling and reasoning support is needed for the early phase. An outline of the i^* framework is given as an example of a step in this direction. Meeting scheduling is used as a domain example.*

1 Introduction

Requirements engineering (RE) is receiving increasing attention as it is generally acknowledged that the early stages of the system development life cycle are crucial to the successful development and subsequent deployment and ongoing evolution of the system. As computer systems play increasingly important roles in organizations, it seems there is a need to pay more attention to the early stages of requirements engineering itself (e.g., [6]).

Much of requirements engineering research has taken as starting point the initial requirements statements, which express customer's wishes about what the system should do. Initial requirements are often ambiguous, incomplete, inconsistent, and usually expressed informally. Many requirements languages and frameworks have been proposed for helping make requirements precise, complete, and consistent (e.g., [4] [19] [13] [15]). Modelling techniques (from boxes-and-arrows diagrams to logical formalisms) with varying degrees of analytical support are offered to assist requirements engineers in these tasks. The objective, in these "late-phase" requirements engineering tasks, is to produce a requirements document to pass on ("downstream") to the developers, so that the resulting system would be adequately specified and constrained, often in a contractual setting.

Considerably less attention has been given to supporting the activities that *precede* the formulation of the initial requirements. These "early-phase" requirements engineering activities include those that consider how the intended system would meet organizational goals, why the system is needed, what alternatives might exist, what the implications of the alternatives are for various stakeholders, and how the stakeholders' interests and concerns might be addressed. The emphasis here is on understanding the "whys" that underlie system requirements [37], rather than on the precise and detailed specification of "what" the system should do.

This earlier phase of the requirements process can be just as important as that of refining initial requirements to a requirements specification, at least for the following reasons:

- System development involves a great many assumptions about the embedding environment and task domain. As discovered in empirical studies (e.g., [11]), poor understanding of the domain is a primary cause of project failure. To have a deep understanding about a domain, one needs to understand the interests and priorities and abilities of various actors and players, in addition to having a good grasp of the domain concepts and facts.

- Users need help in coming up with initial requirements in the first place. As technical systems increase in diversity and complexity, the number of technical alternatives and organizational configurations made possible by them constitute a vast space of options. A systematic framework is needed to help developers understand what users want and to help users understand what technical systems can do. Many systems that are technically sound have failed to address real needs (e.g., [21]).

- Systems personnel are increasingly expected to contribute to business process redesign. Instead of automating well-established business processes, systems are now viewed as "enablers" for innovative business solutions (e.g., [22]). More than ever before, requirements engineers need to relate systems to business and organizational objectives.

- Dealing with change is one of the major problems facing software engineering today. Having a representation of organizational issues and rationales in requirements models would allow software changes to be traced all the way to the originating source – the organizational changes that leads to requirements changes [18].

- Having well-organized bodies of organizational and strategic knowledge would allow such knowledge to be shared across domains at this high level, deepening understanding about relationships among domains. This would also facilitate the sharing and reuse of software (and other types of knowledge) across these domains.

- As more and more systems in organizations interconnect and interoperate, it is increasingly important to understand how systems cooperate (with each other and with human agents) to contribute to organizational goals. Early phase requirements models that deal with organizational goals and stakeholder interests cut across multiple systems and can provide a view of the cooperation among systems within an organizational context.

Support for early-phase RE. Early-phase RE activities have traditionally been done informally, and without much tool support. As the complexity of the problem domain increases, it is evident that tool support will be needed to leverage the efforts of the requirements engineer. A considerable body of knowledge would be built up during early-phase RE. This knowledge would be used to supporting reasoning about organizational objectives, system-and-environment alternatives, implications for stakeholders, etc. It is important to retain and maintain this body of knowledge in order to guide system development, and to deal with change throughout the system's life time.

A number of frameworks have been proposed to represent knowledge and to support reasoning in requirements engineering (e.g., [19] [13] [27] [17] [12] [5] [35]). However, these frameworks have not distinguished early-phase from late-phase RE. The question then is: Are there modelling and reasoning support needs that are especially relevant to early-phase RE? If there are specific needs, can these be met by adapting existing frameworks?

Most existing requirements techniques and frameworks are intended more for the later phase of requirements engineering, which focuses on completeness, consistency, and automated verification of requirements. In contrast, the early phase aims to model and analyze stakeholder interests and how they might be addressed, or compromised, by various system-and-environment alternatives. In this paper it is argued that, because early-phase RE activities have objectives and presuppositions that are different from those of the late phase, it would be appropriate to provide different modelling and reasoning support for the two phases. Nevertheless, a number of recently developed RE techniques, such as agent- and goal-oriented techniques (e.g., [16] [14] [17] [12] [7]) are relevant, and may be adapted for early-phase RE.

The recently proposed *i** framework [39] is used in this paper as an example to illustrate the kinds of modelling features and reasoning capabilities that might be appropriate for early-phase requirements engineering. It introduces an ontology and reasoning support features that are substantially different from those intended for late-phase RE (e.g., as developed in [15]).

Section 2 reviews the *i** framework and outlines some of its features, using meeting scheduling as a domain example. Section 3 discusses, in light of the experience of developing *i**, the modelling and support requirements for early-phase requirements engineering. Section 4 reviews related work. Section 5 draws some conclusions from the discussions and identifies future work.

2 The *i** modelling framework for early-phase requirements engineering

The *i** framework[1] was developed for modelling and reasoning about organizational environments and their information systems [39]. It consists of two main modelling components. The Strategic Dependency (SD) model is used to describe the dependency relationships among various actors in an organizational context. The Strategic Rationale (SR) model is used to describe stakeholder interests and concerns, and how they might be addressed by various configurations of systems and environments. The framework builds on a knowledge representation approach to information system development [27]. This section offers an overview of some of the features of *i**, using primarily a graphical representation. A more formal presentation of the framework appears in [39]. The *i** framework has also been applied to business process modelling and redesign [41] and to software process modelling [38].

The central concept in *i** is that of the intentional actor [36]. Organizational actors are viewed as having intentional properties such as goals, beliefs, abilities, and commitments. Actors depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished. By depending on others, an actor may be able to achieve goals that are difficult or impossible to achieve on its own. On the other hand, an actor becomes vulnerable if the depended-on actors do not deliver. Actors are strategic in the sense that they are concerned about opportunities and vulnerabilities, and seek rearrangements of their environments that would better serve their interests. [2]

2.1 Modelling the embedding of systems in organizational environments – the Strategic Dependency model

Consider a computer-based meeting scheduler for supporting the setting up of meetings[3]. The requirements might state that for each meeting request, the meeting

[1] The name *i** refers to the notion of distributed intentionality which underlies the framework.

[2] An early version of the framework was presented in [36].

[3] The example used in this paper is a simplified version of the one provided in [34].

scheduler should try to determine a meeting date and location so that most of the intended participants will participate effectively. The system would find dates and locations that are as convenient as possible. The meeting initiator would ask all potential participants for information about their availability to meet during a date range, based on their personal agendas. This includes an exclusion set – dates on which a participant cannot attend the meeting, and a preference set – dates preferred by the participant for the meeting. The meeting scheduler comes up with a proposed date. The date must not be one of the exclusion dates, and should ideally belong to as many preference sets as possible. Participants would agree to a meeting date once an acceptable date has been found.

Many requirements engineering frameworks and techniques have been developed to help refine this kind of requirements statements to achieve better precision, completeness, and consistency. However, to develop systems that will truly meet the real needs of an organization, one often needs to have a deeper understanding of how the system is embedded in the organizational environment.

For example, the requirements engineer, before proceeding to refine the initial requirements, might do well to inquire:

- Why is it necessary to schedule meetings ahead of time?
- Why does the meeting initiator need to ask participants for exclusion dates and preferred dates?
- Why is a computer-based meeting scheduler desired? And whose interests does it serve?
- Is confirmation via the computer-based scheduler sufficient? If not, why not?
- Are important participants treated differently? If so, why?

Most requirements models are ill-equipped to help answer such questions. They tend to focus on the "what" rather than the "why". Having answers to these "why" questions are important not only to help develop successful systems in the first instance, but also to facilitate the development of cooperation with other systems (e.g., project management systems and other team coordination "groupware" for which meeting information may be relevant), as well as the ongoing evolution of these systems.

To provide a deeper level of understanding about how the proposed meeting scheduler might be embedded in the organizational environment, the Strategic Dependency model focuses on the *intentional* relationships among organizational actors. By noting the dependencies that actors have on one another, one can obtain a better understanding of the "whys".

Consider first the organizational configuration before the proposed system is introduced (Figure 1). The meeting initiator *depends* on meeting participants `p` to attend meeting `m`. If some participant does not attend the meeting, the meeting initiator may fail to achieve some goal (not made

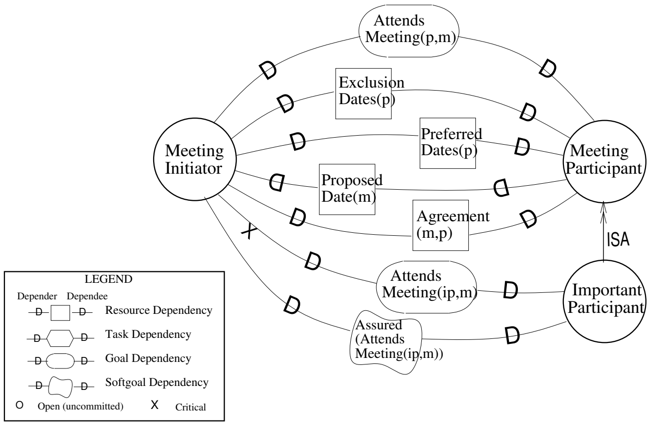

Figure 1: Strategic Dependency model for meeting scheduling, without computer-based scheduler

explicit in the SD model), or at least not succeed to the degree desired. This is the reason for wanting to schedule the meeting in advance. To schedule meetings, the initiator depends on participants to provide information about their availability – in terms of a set of exclusion dates and preferred dates. (For simplicity, we do not separately consider time of day or location.) To arrive at an agreeable date, participants depend on the initiator for date proposals. Once proposed, the initiator depends on participants to indicate whether they agree with the date. For important participants, the meeting initiator depends critically (marked with an "X" in the graphical notation) on their attendance, and thus also on their assurance that they will attend.

Dependency types are used to differentiate among the kinds of relationships between depender and dependee, involving different types of freedom and constraint. The meeting initiator's dependency on participant's attendance at the meeting (`AttendsMeeting(p,m)`) is a *goal dependency*. It is up to the participant how to attain that goal. An agreement on a proposed date `Agreement(m,p)` is modelled as a *resource dependency*. This means that the participant is expected only to give an agreement. If there is no agreement, it is the initiator who has to find other dates (do problem solving). For an important participant, the initiator critically depends on that participant's presence. The initiator wants the latter's attendance to be assured (`Assured[AttendsMeeting(p,m)]`). This is modelled as a *softgoal dependency*. It is up to the depender to decide what measures are enough for him to be assured, e.g., a telephone confirmation. These types of relationships cannot be expressed or distinguished in non-intentional models that are used in most existing requirements modelling frameworks.

Figure 2 shows an SD model of the meeting scheduling setting with a computer-based meeting scheduler. The meeting initiator delegates much of the work of meeting scheduling to the meeting scheduler. The initiator no longer needs to be bothered with collecting availability information from participants, or to obtain agreements about proposed dates from them. The meeting scheduler also determines what are the acceptable dates, given the avail-

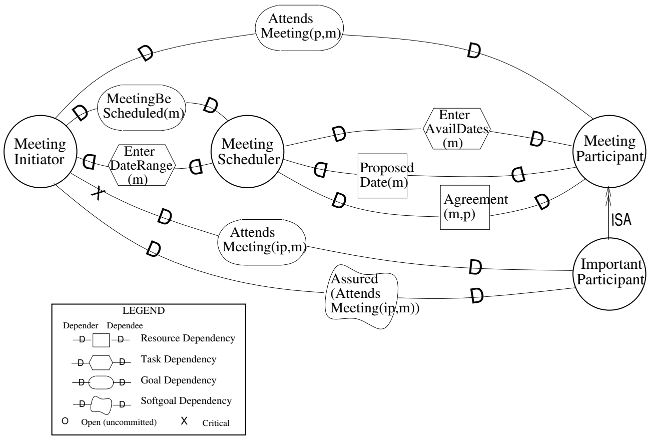

Figure 2: Strategic Dependency model for meeting scheduling with computer-based scheduler

ability information. The meeting initiator does not care how the scheduler does this, as longer as the acceptable dates are found. This is reflected in the *goal dependency* of `MeetingBeScheduled` from the initiator to the scheduler. The scheduler expects the meeting initiator to enter the date range by following a specific procedure. This is modelled via a *task dependency*.

Note that it is still the meeting initiator who *depends* on participants to attend the meeting. It is the meeting initiator (not the meeting scheduler) who has a stake in having participants attend the meeting. Assurance from important participants that they will attend the meeting is therefore not delegated to the scheduler, but retained as a dependency from meeting initiator to important participant.

The SD model models the meeting scheduling process in terms of intentional relationships among agents, instead of the flow of entities among activities. This allows analysis of opportunity and vulnerability. For example, the ability of a computer-based meeting scheduler to achieve the goal of `MeetingBeScheduled` represents an opportunity for the meeting initiator not to have to achieve this goal himself. On the other hand, the meeting initiator would become vulnerable to the failure of the meeting scheduler in achieving this goal.

2.2 Modelling stakeholder interests and rationales – the Strategic Rationale model

The Strategic Dependency model provides one level of abstraction for describing organizational environments and their embedded information systems. It shows external (but nevertheless intentional) relationships among actors, while hiding the intentional constructs within each actor. As illustrated in the preceding section, the SD model can be useful in helping understand organizational and systems configurations as they exist, or as proposed new configurations.

During early-phase RE, however, one would also like to have more explicit representation and reasoning about actors' interests, and how these interests might be addressed or impacted by different system-and-environment configurations – existing or proposed.

In the i^* framework, the Strategic Rationale model provides a more detailed level of modelling by looking "inside" actors to model internal intentional relationships. Intentional elements (goals, tasks, resources, and softgoals) appear in the SR model not only as external dependencies, but also as internal elements linked by means-ends relationships and task-decompositions (Figure 3). The SR model in Figure 3 thus elaborates on the relationships between the meeting initiator and meeting participant as depicted in the SD model of Figure 1.

For example, for the meeting initiator, an internal goal is that of `MeetingBeScheduled`. This goal can be met (represented via a means-ends link) by scheduling meetings in a certain way, consisting of (represented via task-decomposition links): obtaining availability dates from participants, finding a suitable date (and time) slot, proposing a meeting date, and obtaining agreement from the participants.

These elements of the `ScheduleMeeting` task are represented as subgoals, subtasks, or resources depending on the type of freedom of choice as to how to accomplish them (analogous to the SD model). Thus `FindSuitableSlot`, being a subgoal, indicates that it can be achieved in different ways. On the other hand, `ObtainAvailDates` and `ObtainAgreement` refer to specific ways of accomplishing these tasks. Similarly, `MeetingBeScheduled`, being represented as a goal, indicates that the meeting initiator believes that there can be more than one way to achieve it (to be discussed in section 2.4, Figure 4).

`MeetingBeScheduled` is itself an element of the higher-level task of organizing a meeting. Other subgoals under that task might include equipment be ordered, or that reminders be sent (not shown). This task has two additional elements which specify that the organizing of meetings should be done quickly and not involve inordinate amounts of effort. These qualitative criteria are modelled as softgoals. These would be used to evaluate (and also to help identify) alternative means for achieving ends. In this example, we note that the existing way of scheduling meetings is viewed as contributing negatively towards the `Quick` and `LowEffort` softgoals.

On the side of the meeting participants, they are expected to do their part in arranging the meeting, and then to attend the meeting. For the participant, arranging the meeting consists primarily of arriving at an agreeable date. This requires them to supply availability information to the meeting initiator, and then to agree to the proposed dates. Participants want selected meeting times to be convenient, and want meeting arranging activities not to present too many interruptions.

The SR model thus provides a way of modelling stakeholder interests, and how they might be met, and the stakeholders evaluation of various alternatives with respect to their interests. *Task-decomposition links* provide a hierarchical description of intentional elements that make up a *routine*. The *means-ends links* in the SR provides understanding about *why* an actor would engage in some tasks,

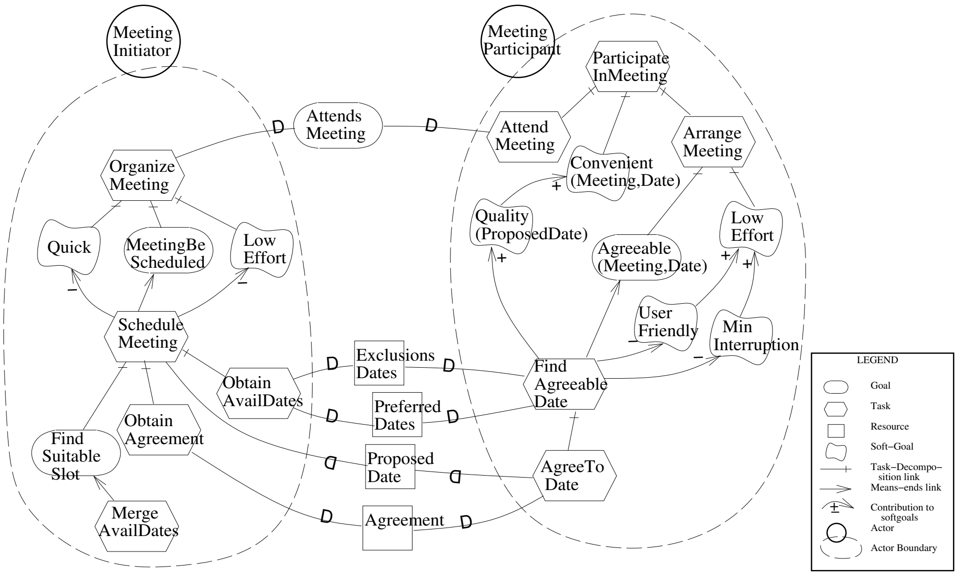

Figure 3: A Strategic Rationale model for meeting scheduling, before considering computer-based meeting scheduler

pursue a goal, need a resource, or want a softgoal. From the softgoals, one can tell *why* one alternative may be chosen over others. For example, availability information in the form of exclusion sets and preferred sets are collected so as to minimize the number of rounds and thus to minimize interruption to participants.

2.3 Supporting analysis during early-phase RE

While requirements analysis traditionally aims to identify and eliminate incompleteness, inconsistencies, and ambiguities in requirements specifications, the emphasis in early-phase RE is instead on helping stakeholders gain better understanding of the various possibilities for using information systems in their organization, and of the implications of different alternatives. The i^* models offer a number of levels of analysis, in terms of *ability, workability, viability* and *believability*. These are detailed in [39] and briefly outlined here.

When a meeting initiator has a routine to organize a meeting, we say that he is *able* to organize a meeting. An actor who is able to organize one type of meeting (say, a project group meeting) is not necessarily able to organize another type of meeting (e.g., the annual general meeting for the corporation). One needs to know what subtask, subgoals, resources are required, and what softgoals are pertinent.

Given a routine, one can analyze it for *workability* and *viability*. Organizing meeting is workable if there is a workable routine for doing so. To determine workability, one needs to look at the workability of each element – for example, that the meeting initiator can obtaining availability information from participants, can find agreeable dates, and can obtain agreements from participants. If the workability of an element cannot be judged primitively by the actor, then it needs to be further reduced. If the subgoal `FindSuitableSlot` is not primitively workable, it will need to be elaborated in terms of a particular way for achieving it. For example, one possible means for achieving it is to do an intersection of the availability information from all participants. If this task is judged to be workable, then the `FindSuitableSlot` goal node would be workable. A task can be workable by way of external dependencies on other actors. The workability of `ObtainAvailDates` and `ObtainAgreement` are evaluated in terms of the workability of the *commitment* of meeting participants to provides availability information and agreement. A more detailed characterization of these concepts are given in [39].

A routine that is workable is not necessarily viable. Although computing intersection of time slots by hand is possible, it is slow and error-prone. Potentially good slots may be missed. When softgoals are not satisficed, the routine is not viable. Note that a routine which is not viable from one actor's perspective may be viable from another actor's perspective. For example, the existing way of arranging for meetings may be viable for participants, if the resulting meeting dates are convenient, and the meeting arrangement efforts do not involve too much interruption of work.

The assessment of workability and viability is based on many beliefs and assumptions. These can be provided as justifications for the assessment. The *believability* of the rationale network can be analyzed by checking the network of justifications for the beliefs. For example, the argument that "finding agreeable dates by merging available dates" is workable may be justified with the assertion that the meeting initiator has been doing it this way for years, and it works. The belief that meeting participants will supply availability information and agree to meeting dates may be

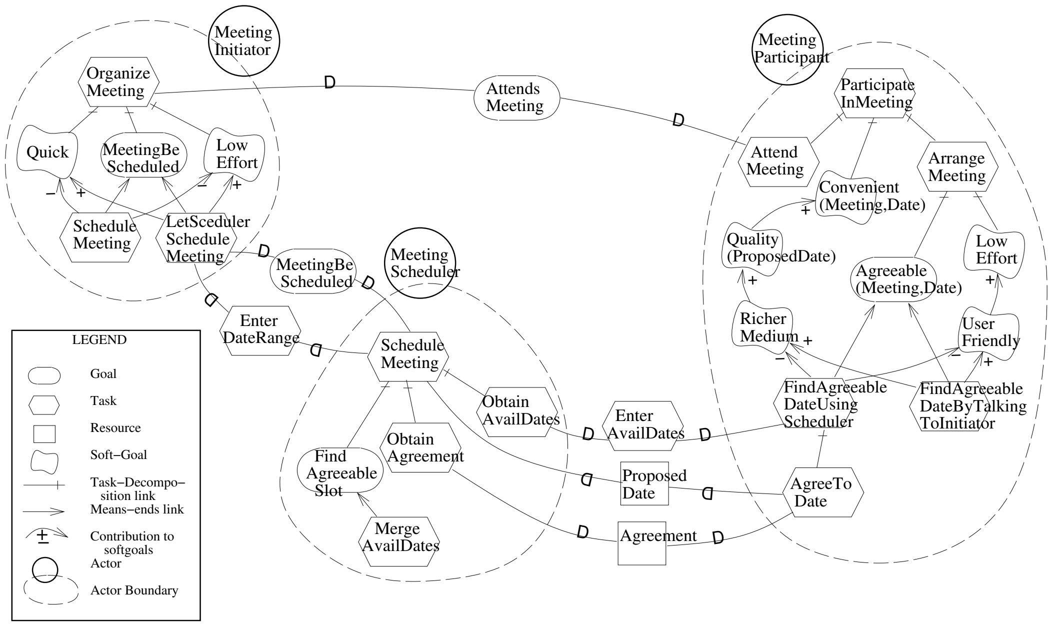

Figure 4: Strategic Rationale model for a computer-supported meeting scheduling configuration

justified by the belief that it is in their own interests to do so (e.g., programmers who want their code to pass a review). The evaluation of these goal graphs (or justification networks) is supported by graph propagation algorithms following a qualitative reasoning framework [8] [42].

2.4 Supporting design during early-phase RE

During early-phase RE, the requirements engineer assists stakeholders in identifying system-and-environment configurations that meet their needs. This is a process of design on a higher level than the design of the technical system per se. In analysis, alternatives are evaluated with respect to goals. In design, goals can be used to help generate potential solutions systematically.

In ***i****, the SR model allows us to *raise* ability, workability, and viability as *issues* that need to be addressed. Using means-ends reasoning, these issues can be *addressed* systematically, resulting in new configurations that are then to be evaluated and compared. Means-ends rules that encode knowhow in the domain can be used to suggest possible alternatives. Issues and stakeholders that are *cross-impacted* may be discovered during this process, and can be raised so that trade-offs can be made. Issues are *settled* when they are deemed to adequately addressed by stakeholders. Once settled, one can then proceed from the descriptive model of the ***i**** framework to a prescriptive model that would serve as the requirements specification for systems development.[4] Believability can also be raised as an issue, so that assumptions would be justified.

In analyzing the SR model of Figure 3, it is found that the meeting initiator is dissatisfied with the amount of effort needed to schedule a meeting, and how quickly a meeting can be scheduled. These are raised as the issues `Quick[MeetingScheduling]` and `LowEffort[MeetingScheduling]`.

Since the meeting initiator's existing routine for scheduling meetings is deemed unviable, one would need to look for new routines. This is done by raising the meeting initiator's ability to schedule meetings as an issue. To address this issue, one could try to come up with solutions without special assistance, or one could look up *rules* (in a knowledge base) that may be applicable. Suppose a rule is found whose *purpose* is `MeetingBeScheduled` and whose *how* attribute is `LetSchedulerScheduleMeeting`.

```
Class  CanLetSchedulerScheduleMeeting IN Rule WITH
  purpose
      ms:   MeetingBeScheduled
  how
      ssm: LetSchedulerScheduleMeeting
  applicabilityCond
      platform: HasAppropriatePlatform(team,
                              platform,scheduler)
  END
```

This represents knowledge that the initiator has about software scheduler systems, their abilities, and their platform requirements. The rule helps discover that the meeting initiator can delegate the subgoal of meeting scheduling to the (computer-based) meeting scheduler. This constitutes a routine for the meeting initiator.

Using a meeting scheduler, however, requires partici-

[4] One approach to this is described in [40].

pants to enter availability information in a particular format. This is modelled as a *task dependency* on participants (an SD link). A routine that provides for this is sought in the participant. Again, rules may be used to assist in this search.

When new configurations are proposed, they may bring in additional issues. The new alternatives may have associated softgoals. The discovery of these softgoals can also be assisted with means-ends rules. For example, using computer-based meeting scheduling may be discovered to be negative in terms of medium richness and user-friendliness. These in turn have implications for the effort involved for the participant, and the quality of the proposed dates. These newly raised issues also need to be addressed. Once new routines have been identified, they are analyzed for workability and viability. Further routines are searched for until workable and viable ones are found.

3 The modelling and reasoning support needs of early-phase RE

In the preceding section, the i^* framework was outlined in order to illustrate the kind of modelling and reasoning support that would be useful during the early phase of requirements engineering. This section summarizes and discusses these modelling and support needs in more general terms, drawing from the experience of the i^* framework.

Knowledge representation and reasoning. Although the example in the preceding section relies primarily on informal graphical notations, it is clear that a realistically-sized application domain would involve large numbers of concepts and relationships. A more formal knowledge representation scheme would be needed to support modelling, analysis, and design activities. Maintaining a knowledge base of the knowledge collected and used during early-phase RE is also crucial in order to reap benefits for supporting ongoing evolution (e.g., [8]), and for reuse across related domains.

Many of the knowledge-based techniques developed for other phases of software engineering are also applicable here. For example, knowledge structuring mechanisms such as classification, generalization, aggregation, and time [20] are equally relevant in early-phase as in late-phase RE. On the other hand, early-phase RE has certain needs that are quite distinct from late-phase RE.

Degree of formality. While representing knowledge formally has the advantage of amenability to computer-based tool support, the nature of the early-phase suggests that formality should be used judiciously. The early-phase RE process is likely to be a highly interactive one, with the stakeholders as the source of information as well as the decision maker. The requirements engineer acts primarily in a supporting role. The degree of formality for a support framework therefore needs to reflect this relationship. Use of knowledge representation can facilitate knowledge management and reasoning. However, one should not try to over formalize, as one may compromise the style of reasoning needed.

One approach is to introduce weaker constructs, such as softgoals, which requires judgemental inputs from time to time in the reasoning process, but which can be structured and managed nonetheless within the overall knowledge base [7] [39]. The notion of softgoal draws on the concept of satisficing [33], which refers to finding solutions that are "good enough".

Incorporating intentionality. One of the key needs in dealing with the subject matter in the early phase seems to be the incorporation of the concept of the intentional actor into the ontology. Without intentional concepts such as goals, one cannot easily deal with the "why" dimension in requirements.

A number of requirements engineering frameworks have introduced goal-oriented and agent-oriented techniques (e.g., [16] [14] [17] [12] [7]). In adapting these techniques for early-phase RE, one needs to recognize that the focus during the early phase is on *modelling* (i.e., describing) the intentionality of the stakeholders and players in the organizational environment. When new alternatives are being sought (the "design" component in early phase RE), it is the intentionality of the stakeholders that are being exercised. The requirements engineer is helping stakeholders find solutions to *their* problems. The decisions rests with the stakeholders.

In most goal-oriented frameworks in RE, the intentionality is assumed to be under the control of the requirements engineer. The requirements engineer manipulates the goals, and makes decisions on appropriate solutions to these goals. This may be appropriate for late-phase RE, but not for the early phase.

By the end of the early-phase, the stakeholders would have made the major decisions that affect their strategic interests. Requirements engineers and developers can then be given the responsibility to fill in the details and to realize the system.

One consequence of the early/late phase distinction is that intentionality is harder to extract and incorporate into a model in the early phase than in the late phase. Stakeholder interests and concerns are typically not readily accessible. The approach adopted in i^* is to introduce the notion of intentional dependencies to provide a level of abstraction that hides the internal intentional contents of an actor. The Strategic Dependency model provides a useful characterization of the relationships among actors that is at an intentional level (as opposed to non-intentional activities and flows), without requiring the modeller to know much about the actors' internal intentional dispositions. Only when one needs to reason about alternative configurations would one need to make explicit the goals and criteria for such deliberations (in the Strategic Rationale model). Even here, the model of internal intentionality is not assumed to be complete. The model typically contains only those concerns that are voiced by the stakeholders in order for them to achieve the changes they desire.

Multi-lateral intentional relationships. In modelling the embedding of a system in organizational environments, it is necessary to describe dependencies that the system has on its environment (human agents and possibly other systems), as well as the latter's dependencies on the system. When the system does not live up to the expectations of agents in its environment, the latter may fail to achieve certain goals. The reverse can also happen. During early-phase RE, one needs to reason about opportunities and vulnerabilities from both perspectives. Both the system and its environment are usually open to redesign, within limits. When opportunities or vulnerabilities are discovered, further changes can be introduced on either side to take advantage of them or to mitigate against them. A modelling framework for the early-phase thus needs to be able to express multi-lateral intentional relationships and to support reasoning about their consequences.

In most requirements frameworks, the requirements models are interpreted prescriptively. They state what a system is supposed to do. This is appropriate for late-phase RE. Requirements documents are often used in contractual settings – developers are obliged to design the systems in order to meet the specifications. Once the early-phase decisions have settled, a conversion from the multi-lateral dependency model to a unilateral prescriptive model for the late-phase can be made.

Distributed intentionality. Another distinctive feature of the early-phase subject matter is that the multiple actors in the domain all have their own intentionality. Actors exercise intentionality (e.g., they pursue goals) in the course of their daily routines. Actors have multiple, sometimes conflicting, sometimes complementary goals. The introduction of a computer system may make certain goals easier to achieve and others harder to achieve, thus perturbing the network of strategic dependencies. Different system-and-environment configurations can therefore be seen as different ways of re-distributing the pattern of intentionality[5]. The boundaries may shift (the responsibility for achieving certain goals may be delegated from some agents to other agents, some of which may be computer systems), but the actors remain intentional. The process of system-and-environment redesign does not solve all the problems (i.e., does not (completely) reduce intentional elements, such as goals, to non-intentional elements, such as actions). It merely rearranges the terrain in which problems appear and need to be addressed.

In contrast, in late-phase RE and in the rest of system development, one does attempt to fully reduce goals to implementable actions.

Means-ends reasoning. In order to model and support reasoning about "why", and to help come up with alternative solutions, some form of means-ends reasoning would appear necessary. However, a relatively weaker form of reasoning than customarily used in goal-oriented frameworks is needed. This is because of the higher degree of incompleteness in early phase RE. The emphasis is on modelling stakeholders' rationales. Alternative solutions may be put forth as suggestions, but it is the stakeholders who decide. The modelling may proceed both "upwards" and "downwards" (from means to ends or vice versa). There is no definitive "top" (since there may always be some higher goal) nor "bottom" (since there is no attempt to purge intentionality entirely). It is the stakeholders' decision as to when the issues have been adequately explored and a sufficiently satisfactory solution found.

[5] Hence the name *i**.

The type of reasoning support desired is therefore closer to those developed in issue-based information systems, argumentation frameworks, and design rationales (e.g., [10] [30] [26] [25]). The i^* approach is an adaptation of a framework developed for dealing with non-functional requirements [7], which draws on these earlier frameworks.

Organizational actors. In modelling organizational environments, a richer notion of actor is needed. i^* differentiates actors into agents, roles, and positions [39]. In late-phase requirements engineering, where the focus is on specifying behaviours rather than intentional relationships, such distinctions may not be as significant. Viewpoints has been recognized as an important topic in requirements engineering (e.g., [29]). In the early phase, the need to treat multiple viewpoints involving complex relationships among various types of actors is even more important.

4 Related work

In the requirements modelling area, the need to model the environment is well recognized (e.g., [4] [19] [23]). Organization and enterprise models have been developed in the areas of organizational computing (e.g., [1]) and enterprise integration (e.g., [9]). However, few of these models have considered the intentional, strategic aspects of actors. Their focus has primarily been on activities and entities rather than on goals and rationales (the "what" rather than the "why").

A number of requirements engineering frameworks have introduced concepts of agents or actors, and employ goal-oriented techniques. The framework of [5] uses multiple models to model actors, objectives, subject concepts and requirements separately, and is close in spirit to the i^* framework in many ways. The WinWin framework of [2] identifies stakeholder interests and links them to quality requirements. The notion of inquiry cycle in [31] is closely related to the early-phase RE notion, but takes a scenarios approach. The KAOS framework [12] [35] for requirements acquisition employs the notions of goals and agents, and provides a methodology for obtaining requirements specifications from global organizational goals.

However, these frameworks do not distinguish between the needs of early-phase vs. late-phase RE. For example, most of them assume a global perspective on goals, which are reduced, by requirements engineers, in a primarily top-down fashion, fully to actions. These may be contrasted with the notion of distributed intentionality in i^*, where agents are assumed to be strategic, whose intentionality are only partially revealed, who are concerned about opportunities and vulnerabilities, and who seek to advance or protect their strategic interests by restructuring intentional relationships.

5 Conclusions

Understanding "why" has been considered an important part of requirements engineering since its early days [32]. Frameworks and techniques to explicitly support the modelling of and reasoning about agents' goals and rationales have recently been developed in RE. In this paper, it was argued that making a distinction between early-phase and late-phase RE could help clarify the ways in which these concepts and techniques could be applied to different RE activities.

The i^* framework was given as an example in which agent- and goal-oriented concepts and techniques were adapted to address some of the special needs of early-phase RE.

The proposal to use a modelling framework tailored specifically to early-phase RE and a separate framework for late-phase RE implies that a linkage between the two kinds of framework is needed [40]. As with other phases in the software development life cycle, the relationship between early and late phase RE is not strictly sequential or even temporal. Each phase generates and draw on a certain kind of knowledge, which needs to be maintained throughout the life cycle for maximum benefit [24] [20] [28]. The application of knowledge-based techniques to early-phase RE could potentially bring about a more systematic approach to this often *ad hoc*, under-supported phase of system development.

Preliminary assessments of the usefulness of i^* modelling in a real setting have been positive [3]. Supporting tools and usage methodologies are being developed in an on-going project [42].

Acknowledgments

The author gratefully acknowledges the many helpful suggestions from anonymous referees, Eric Dubois, Brian Nixon, and Lawrence Chung, as well as on-going guidance from John Mylopoulos, and financial support from the Information Technology Research Centre of Ontario, and the Natural Sciences and Engineering Research Council of Canada.

References

[1] A. J. C. Blythe, J. Chudge, J.E. Dobson and M.R. Strens, ORDIT: a new methodology to assist in theprocess of eliciting and modelling organizational requirements. *Proc. Conference on Organizational Computing Systems*, Milpitas CA, 1993. pp. 216–227.

[2] B. Boehm and H. In, Aids for Identifying Conflicts Among Quality Requirements, *IEEE Software*, March 1996.

[3] L. Briand, W. Melo, C. Seaman and V. Basili, Characterizing and Assessing a Large-Scale Software Maintenance Organization, *Proc. 17th Int. Conf. Software Engineering.* Seattle, WA. 1995.

[4] J. A. Bubenko, Information Modeling in the Context of System Development, *Proc. IFIP,* 1980, pp. 395-411.

[5] J. A. Bubenko, Extending the Scope of Information Modeling, *Proc. 4th Int. Workshop on the Deductive Approach to Information Systems and Databases,* Lloret-Costa Brava, Catalonia, Sept. 20-22, 1993, pp. 73-98.

[6] J. A. Bubenko, Challenges in Requirements Engineering, *Proc. 2nd IEEE Int. Symposium on Requirements Engineering*, York, England, March 1995, pp. 160-162.

[7] K. L. Chung, *Representing and Using Non-Functional Requirements for Information System Development: A Process-Oriented Approach*, Ph.D. Thesis, also Tech. Rpt. DKBS-TR-93-1, Dept. of Comp. Sci., Univ. of Toronto, June 1993.

[8] L. Chung, B. Nixon and E. Yu, Using Non-Functional Requirements to Systematically Support Change, *2nd IEEE Int. Symp. on Requirements Engineering (RE'95)*, York, England, March 1995.

[9] *CIMOSA – Open Systems Architecture for CIM*, ESPRIT Consortium AMICE, Springer-Verlag 1993.

[10] J. Conklin and M. L. Begeman, gIBIS: A Hypertext Tool for Explanatory Policy Discussions, *ACM Transactions on Office Information Systems*, 6(4), 1988, pp. 303-331.

[11] B. Curtis, H. Krasner and N. Iscoe, A Field Study of the Software Design Process for Large Systems, *Communications of the ACM*, 31(11), 1988, pp. 1268-1287.

[12] A. Dardenne, A. van Lamsweerde and S. Fickas, Goal-Directed Requirements Acquisition, *Science of Computer Programming*, 20, 1993, pp. 3-50.

[13] E. Dubois, J. Hagelstein, E. Lahou, F. Ponsaert and A. Rifaut, A Knowledge Representation Language for Requirements Engineering, *Proc. IEEE*, 74 (10), Oct. 1986, pp. 1431 –1444.

[14] E. Dubois, A Logic of Action for Supporting Goal-Oriented Elaborations of Requirements, *Proc. 5th International Workshop on Software Specification and Design*, Pittsburgh, PA, 1989, pp. 160-168.

[15] Ph. Du Bois, *The Albert II Language – On the Design and the Use of a Formal Specification Language for Requirements Analysis*, Ph.D. Thesis, Department of Computer Science, University of Namur, 1995.

[16] M. S. Feather, Language Support for the Specification and Development of Composite Systems, *ACM Trans. Prog. Lang. and Sys.* 9, 2, April 1987, pp. 198-234.

[17] S. Fickas and R. Helm, Knowledge Representation and Reasoning in the Design of Composite Systems, *IEEE Trans. Soft. Eng.*, 18, 6, June 1992, pp. 470-482.

[18] O.C.Z. Gotel and A.C.W. Finkelstein, An Analysis of the Requirements Traceability Problem, *Proc. IEEE Int. Conf. on Requirements Engineering*, Colorado Springs, April 1994, pp. 94-101.

[19] S. J. Greenspan, J. Mylopoulos, and A. Borgida, Capturing More World Knowledge in the Requirements Specification, *Proc. Int. Conf. on Software Eng.*, Tokyo, 1982.

[20] S. J. Greenspan, J. Mylopoulos and A. Borgida, On Formal Requirements Modeling Languages: RML Revisited, (invited plenary talk), *Proc. 16th Int. Conf. Software Engineering*, May 16-21 1994, Sorrento, Italy, pp. 135-147.

[21] J. Grudin, Why CSCW Applications Fail: Problems in the Design and Evaln of Organizational Interfaces, *Proc. Conference on Computer-Supported Cooperative Work* 1988, pp. 85-93.

[22] M. Hammer and J. Champy, *Reengineering the Corporation: A Manifesto for Business Revolution*, HarperBusiness, 1993.

[23] M. Jackson, *System Development*, Prentice-Hall, 1983.

[24] M. Jarke, J. Mylopoulos, J. W. Schmidt and Y. Vassiliou, DAIDA: An Environment for Evolving Information Systems, *ACM Trans. Information Systems*, vol. 10, no. 1, Jan 1992, pp. 1-50.

[25] J. Lee, *A Decision Rationale Management System: Capturing, Reusing, and Managing the Reasons for Decisions*, Ph.D. thesis, MIT, 1992.

[26] A. MacLean, R. Young, V. Bellotti and T. Moran, Questions, Options, and Criteria: Elements of Design Space Analysis, *Human-Computer Interaction*, vol. 6, 1991, pp. 201-250.

[27] J. Mylopoulos, A. Borgida, M. Jarke and M. Koubarakis, Telos: Representing Knowledge about Information Systems, *ACM Trans. Info. Sys.*, 8 (4), 1991.

[28] J. Mylopoulos, A. Borgida and E. Yu, Representing Software Engineering Knowledge, *Automated Software Engineering*, to appear.

[29] B. Nuseibeh, J. Kramer and A. Finkelstein, Expressing the Relationships Between Multiples Views in Requirements Specification, *Proc. 15th Int. Conf. on Software Engineering*, Baltimore, 1993, pp. 187-196.

[30] C. Potts and G. Bruns, Recording the Reasons for Design Decisions, *Proc. 10th Int. Conf. on Software Engineering* , 1988, pp. 418-427.

[31] C. Potts, K. Takahashi and A. Anton, Inquiry-Based Requirements Analysis, *IEEE Software*, March 1994, pp. 21-32.

[32] D. T. Ross and K. E. Shoman, Structured Analysis for Requirements Definition, *IEEE Trans. Soft. Eng.*, Vol. SE-3, No. 1, Jan. 1977.

[33] H. A. Simon, *The Sciences of the Artificial*, 2nd ed., Cambridge, MA: The MIT Press, 1981.

[34] A. Van Lamsweerde, R. Darimont and Ph. Massonet, The Meeting Scheduler Problem: Preliminary Definition. Copies may be obtained from Prof. Van Lamsweerde, Universite Catholique de Louvain, Unite d'Informatique, Place Sainte-Barbe, 2, B-1348 Louvain-la-Neuve, Belgium. (avl@info.ucl.ac.be)

[35] A. Van Lamsweerde, R. Darimont and Ph. Massonet, Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt, *Proceedings of 2nd IEEE Int. Symposium on Requirements Engineering*, York, England, March 1995, pp. 194-203.

[36] E. Yu, Modelling Organizations for Information Systems Requirements Engineering, *Proceedings of First IEEE Symposium on Requirements Engineering*, San Diego, Calif., January 1993, pp. 34-41.

[37] E. Yu and J. Mylopoulos, Understanding Why in Requirements Engineering – with an Example, *Workshop on System Requirements: Analysis, Management, and Exploitation*, Schloß Dagstuhl, Saarland, Germany, October 4–7, 1994.

[38] E. Yu and J. Mylopoulos, Understanding 'Why' in Software Process Modelling, Analysis, and Design, *Proc. 16th Int. Conf. on Software Engineering*, Sorrento, Italy, May 1994, pp. 159-168.

[39] E. Yu, *Modelling Strategic Relationships for Process Reengineering*, Ph.D. thesis, also Tech. Report DKBS-TR-94-6, Dept. of Computer Science, University of Toronto, 1995.

[40] E. Yu, P. Du Bois, E. Dubois and J. Mylopoulos, From Organization Models to System Requirements – A 'Cooperating Agents' Approach, *Proc. 3rd Int. Conf. on Cooperative Information Systems (CoopIS-95)*, Vienna, Austria, May 1995, pp. 194-204.

[41] E. Yu and J. Mylopoulos, From E-R to 'A-R' – Modelling Strategic Actor Relationships for Business Process Reengineering, *Int. Journal of Intelligent and Cooperative Information Systems*, vol. 4, no. 2 & 3, 1995, pp. 125-144.

[42] E. Yu, J. Mylopoulos and Y. Lesperance, AI Models for Business Process Reengineering, *IEEE Expert*, August 1996, pp. 16-23.