Incorporating System Dynamics Modeling into Goal-oriented Adaptive Requirements Engineering

Yong Du
Department of Computer Science
University of Toronto
10 King's College Road
Toronto, Ontario
Canada M5S 3G4
ydu@cs.toronto.edu

Abstract

Adaptive requirements are system requirements that change either internally, or externally in the environment, where environment includes the users and other systems interacting with the subject system. Modeling and analyzing adaptive requirements are important because change is a ubiquitous phenomenon in social-technical systems. Engineering adaptive requirements is also difficult because many contextual factors need to be taken account of, and some of these factors (e.g., human intentional changes) are hard to quantify. This paper investigates an approach to modeling adaptive requirements by integrating system dynamics modeling into i goal-oriented modeling. We illustrate that each modeling method has its advantages that are difficult or impossible to achieve using the other method, and each has its limitations that can be overcome using the other method. We then propose a goal-oriented adaptive requirements engineering (GARE) approach that integrates desirable features of system dynamics into i*, taking advantage of the goal-oriented modeling initiatives in i* and the built-in constructs in system dynamics for modeling time-based system evolution and adaptation.*

1 Introduction

Modern systems continue to adapt themselves to internal changes of system components, or external changes of the system environment. Physically speaking, the system environment includes the user, usage data, time, location, and other contextual information (Abowd and Mynatt 2000; Jameson 2001; Kobsa 2001). Virtually speaking, social and political atmosphere, psychological and ethical attitude, as well as human mental characteristics, all belong to the system environment (Petak 1981). All these factors continuously change, creating a dynamic environment in which a system is required to operate normally. To design such a system, careful considerations of the evolution of its environment and decisions on how the system adapts to the environmental changes

must be made early in the requirements engineering stage in order to generate a system that is both adaptable and adaptive.

In this paper, we define adaptive requirements as system requirements that change according to the environment, where environment includes both physical context (e.g., user, usage, time, and location) and virtual context (e.g., social architecture, political atmosphere, ethical attitude, and mental characteristics). Adaptive requirements are especially common for social-technical systems where organizational structures, interpersonal relationships, and behavioral decisions constantly change. Misunderstanding of adaptive requirements and inappropriate adaptation to system evolution can lead to asynchronous adaptation (Leplat 1984; Leveson 2004), which can cause potential system failure, hazardous situation in safety-critical systems, and tremendous loss of business values in security-critical systems.

Modeling and analyzing adaptive requirements is difficult due to the complexity of integrating the time factor into the reasoning process. First, it becomes complicated to graphically model the time factor and simulate the adaptation effects when the time factor is combined with other factors (e.g., system evolution over time, or change of human mental characteristics over time). Second, there are different types of system changes based on time, and some of these changes are irregular and difficult to model. Third, the number of cases of system changes and their effects can grow exponentially to a degree where exhaustive analyses of these cases become computationally infeasible, in which situation a proper case generation strategy (e.g., boundary value generation (Du and Hoffman 2004; Ramachandran 2003), and pairwise generation (Cohen, Dalal, Fredman, and Patton 1997; Tai and Lei 2002)) must be carefully selected. In spite of these challenges, system degeneration and adaptation are considered as a process that is predictable and controllable (Leveson 2004; Rasmussen 1997), which means it is not only desirable but also possible to model adaptive requirements; and this work must be performed early in the requirements engineering stage in order to thoroughly understand adaptive requirements before the actual design of an adaptive system.

While there is a good opportunity of applying previous requirements engineering research results to studying adaptive requirements, it must be noted that existing requirements engineering methods, techniques, and processes may not fit properly for adaptive requirements. There has been some recent work on adapting known requirements engineering methods and techniques to adaptive and adaptable systems, including ubiquitous computing (especially context-awareness) (Finkelstein, Savigni, Kappel, Retschitzegger, Schwinger, and Feichtner 2002; Kappel, Proll, Retschitzegger, Schwinger, and Hofer 2001; Niemela and Latvakoski 2004), pervasive computing (Kolos-Mazuryk, Poulisse, and van Eck 2005), adaptive systems (Goldsby and Cheng 2006), and adaptive hypermedia systems (Tsandilas and Schraefel 2004). While constructive recommendations have been proposed in these research attempts, no proposed solution has demonstrated theoretical maturity or evident empirical reliability. Researchers have not arrived at a point where one or more proposals are likely to provide convincing results dealing with adaptive requirements engineering. Furthermore, since adaptive requirements engineering is expected to contribute to multiple research disciplines (e.g., social-technical system, ubiquitous computing, pervasive computing, adaptive system, and adaptive hypermedia system), it is reasonable to believe that one solution does not fit all.

This paper studies the modeling and analysis of adaptive requirements in social-technical systems using two approaches: system dynamics and *i**. System dynamics is a method used to model the dynamic features of large-scale systems. Since its initial proposal in the 1950s, it has been

widely applied to modeling industrial production process, business process, economic cycles, environmental policies and effects, among other physical or social domains. Specifically, the past thirty years have seen tremendous research activities and industrial practices in modeling and analyzing social-technical systems using the system dynamics approach because this approach has prominent advantages of modeling dynamic changes of a system over time.

It is widely acknowledged that human factors play the most crucial role in the operations of social-technical systems, and human behaviors are triggered by various goals. However, system dynamics modeling has the inherited defect that the modeling process is neither actor-oriented nor goal-oriented. As a result, it is not convenient to use system dynamics modeling for illustrating the human objective, desire, intention, belief, trust, hatred, and other mental characteristics.

The *i** modeling framework is both agent-oriented and goal-oriented. It is agent-oriented because it starts by identifying all the actors in the problem domain. It is goal-oriented as actors are modeled and analyzed based on their goals. This modeling framework associates human mental characteristics with their owning actors, making it appropriate for social-technical systems where instability of human behaviors forms the dynamic basis of the systems. However, the *i** modeling framework lacks modeling constructs for representing the time factor and the system changes over time. For example, people adjust their goals frequently, but in *i** models, an actor normally has one or more fixed goals and acts based on those unchanging goals.

This paper proposes *GARE*, a Goal-oriented Adaptive Requirements Engineering approach by integrating the system dynamics method into the *i** goal-oriented modeling framework. We start the effort by investigating the limitations of each method using the strengths of the other method. We then map the model elements between system dynamics and *i** to further compare their strength and weakness. Finally, we incorporate system dynamics into *i** by introducing evolution labels, adaptation links, and evolution label propagation rules. Examples in the drinking water quality control domain are used to demonstrate the modeling and analysis process, and these examples are based on the study of the Walkerton water contamination accident, which happened in the town of Walkerton of Ontario, Canada in May 2000, causing 7 deaths and more than 2300 illness. More information about this accident can be found in (Leveson, Daouk, Dulas, and Marais 2004; O'Connor 2002; Vicente and Christoffersen 2006). Choosing the drinking water quality control system as our case study does not limit *GARE* to safety engineering. The modeling and analysis process applied in this paper can be reused for any safety or security critical systems, as well as other social-technical systems.

Section 2 introduces the system dynamics modeling method and discusses its limitations on modeling goals and other intentional characteristics. Section 3 introduces *i** and investigates its limitations on modeling time-based system evolution and adaptation. Section 4 demonstrates in examples the mapping process between system dynamics model elements and *i** model elements. Section 5 proposes the *GARE* approach by combining system dynamics modeling features into *i**. The last section concludes this study and lists potential future works.

2 System Dynamics

2.1 Overview

System dynamics modeling is a technique originally developed by Jay Forrester in the 1950s. It has been widely used to analyze the dynamic characteristics of large systems, to study the cause and effect relationships between system components, and to simulate adaptations to system evolution.

In the following paragraphs, we introduce the fundamental concepts of system dynamics, which are stock, flow, variable, link, feedback loop, time path, and delay. We explain these concepts using an example model in the drinking water quality control problem domain, as given in Figure 1. Readers can refer to (Forrester 1961; Randers 1980; Richardson and Pugh 1981; Sterman 2000; Sterman 2002) for in-depth explanation of system dynamics.

The most fundamental concepts in system dynamics are stocks and flows. A *stock* represents a container of items whose quantity increases or decreases over time. A *flow* is the rate of movement of items into or out of a stock. An *inflow* is a flow moving items into a stock, and an *outflow* is a flow moving items out of a stock. The *net flow* of a stock is the sum of all inflows minus the sum of all outflows. There are three stocks with in/outflows in Figure 1.

A complicated stock-flow system contains not only stocks and flows, but also variables and links. A *variable* represents a concrete component or state of a system feature. A *link* represents the cause-effect relationship between variables, stocks, and flows. Each link has an *orientation* and a *polarity*. A positive polarity means a change of the source component will lead to a change of the destination component in the same direction. A negative polarity means the change of the source component will lead to a change of the destination component in the opposite direction. The variables and links in Figure 1 illustrate the causes and effects of changes in the drinking water quality control system.

The sequential effects of causal links form *feedback loops*, which are important in understanding system dynamics models. There are three feedback loops in Figure 1. As an example, feedback loop L_1 demonstrates the following series of effects:

1. When the water operator begins his new job, he tries his best to avoid any punishment in order to keep the job. So the water operator has a high level `Fear of Punishment` and a high `Compliance with Quality Control Process`.
2. After a certain amount of time, the water operator's `Fear of Punishment` starts to drop, which will cause his `Compliance with Quality Control Process` to decrease accordingly.
3. Low-level `Compliance with Quality Control Process` will reduce `Effective Water Sampling` and `Effective Chlorination`, both of which will lead to more `Accidents/Incidents`.
4. More `Accidents/Incidents` will increase `Rate of Historical Punishment`, which will increase water operator's `Fear of Punishment`.

We can see that decrease of water operator's `Fear of Punishment` will finally feedback to itself, causing `Fear of Punishment` to increase. This negative feedback loop, indicated by the negative sign (-), acts as a mechanism to bring this stock to an equilibrium state over a long period of system operation. Similarly, L_2 and L_3 are also negative feedback loops having controlling effects on stock `Confidence in Water Quality` and stock `Risk of Drinking Water Infection`. The sign of a feedback loop (either + or -) indicates whether the overall effect of the feedback loop is positive or negative.

Delay is another important and common characteristic of dynamic systems. A process containing a sequence of tasks needs a certain amount of time to finish, and the time for each task can frequently be underestimated, leading to delays in fulfilling a goal. Understanding delays in a dynamic system is crucial to designing effective mechanisms for delivering expected results. Figure 1 shows two delays:

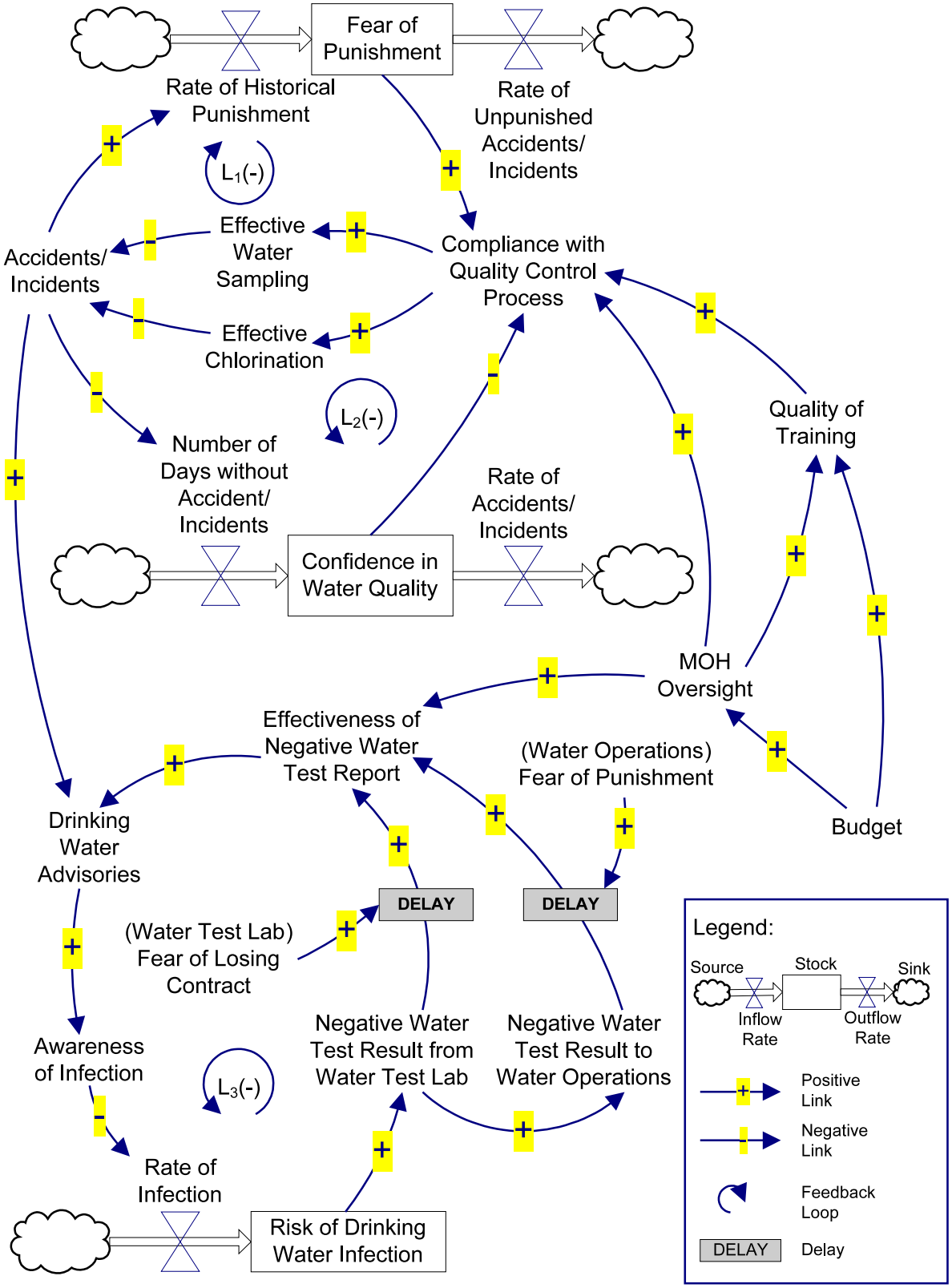

Figure 1. System Dynamics Model - Illustration of Stocks, Flows, Variables, Links, Feedback Loops, and Delays

1. `Water Test Lab` delays `Negative Water Test Report` to `MOH` (Ministry of Health) because it fears that `Water Operations` would terminate the contract.
2. `Water Operations` delays `Negative Water Test Report` to `MOH` due to `Fear of Punishment`.

Furthermore, a `Budget` cut forces `MOH` to loose the `Oversight` on drinking water quality, giving `Water Test Lab` and `Water Operations` opportunities of delaying or even ignoring negative water test report. Delays can have significant effect on a system. In Figure 1, delayed `Negative Water Test Report` leads to delayed `Drinking Water Advisories` and `Awareness of Infection`, causing the effect of the negative feedback loop L_3 to be delayed and `Risk of Drinking Water Infection` to remain high during the delayed period of time. As a matter of fact, in the Walkerton water contamination accident, there was a 5-day delay from when the negative water test result came out to when the report was disclosed to MOH; and the health advisory was issued to the public four days after the infection was detected, leading to widespread fatal illness in the community.

System dynamics modeling concerns the behavior of a system over a long period of time. Systems which are designed to work properly today may function disastrously in the future, so system dynamics modeling aims at building systems that are sustainable. Figure 1 demonstrates that if the value of a variable or the level of a stock changes, there is a sequential effect of that change on other variables and levels. However, the model does not say how the stock or variable changes. Is it a linear increase or exponential decrease? Or, is it an irregular change that does not follow any pattern? *Time path* is a method to depict the changes of a system feature over time. Commonly used time paths include linear time path, exponential time path, goal-seeking time path, and oscillation time path (Randers 1980; Sterman 2000). In practice, time paths are almost always non-linear; instead, they are more likely to be composite time paths consisting of segments of time paths of different types.

System simulation is a commonly available feature in system dynamics (Roberts, Andersen, Deal, and Shaffer 1983; Sterman 1987) and general systems theory (Checkland 1999; Weinberg 2001). Simulation is concerned about performing thorough cause-effect analysis on the system in order to detect any unexpected future states of the system under certain circumstances. In order to predict potential system adaptation, we need to assign initial evolution values to selected system elements, determine cause-effect transition rules from one element to the other elements, and calculate the effects by following cause-effect links. A simulation tool (Richardson and Pugh 1981) can normally automate this process. Theories on simulation using system dynamics and related tool support have been widely proposed and discussed, and commercial applications of system dynamics simulation are also available.

Now that we have introduced the fundamental concepts and benefits of system dynamics, we then discuss the limitations of system dynamics model.

2.2 Limitations

2.2.1 Ambiguous Subject and Object

The first limitation of the system dynamics model is that it is not clear which system component is responsible for each stock or variable. In other words, the subject of each stock or variable is unclear, and we call it the *ambiguous subject* problem. In addition to the ambiguous subject problem, there also exists the *ambiguous object* problem, which means the target of an action or

the destination of a resource is unclear. The model in Figure 1 contains the following subject and/or object ambiguity:

- Who provides `Budget` to whom?
- Who performs `Oversight` on whom?
- Who owns the stock of `Fear of Punishment` from whom?
- Who needs `Negative Water Test Report` from whom?
- Who has `Confidence in Water Quality`?
- Who is responsible for controlling `Risk of Drinking Water Infection`?

To partly solve the ambiguous subject and object problems, we can stipulate a subject and an object for each stock or variable. This tactic has been applied to two variables in Figure 1 to improve understanding of the model. However, this treatment does not fundamentally solve the ambiguity. First, it may not be appropriate to add a subject or object to some stocks or variables (e.g., how to add a subject to stock `Risk of Drinking Water Infection`? And how to add an object to stock `Confidence in Water Quality`?). Second, even if it is possible to assign a subject and an object to all variables and stocks, the responsibilities of each subject are scattered around the model without clear clue that they belong to the same subject. Third, plain sentences are not formal and precise means of modeling relationships. Here we expect to see some simple model illustrating the dependency relationship between a subject and an object, but system dynamics is not able to model such types of dependency relationships.

2.2.2 Limitation on Modeling Intentions

Today's social-technical systems consist of various kinds of *social components* (e.g., individual person, government, business group, agency, and trade organization), each having different goals, beliefs, trust, confidence, and other *intentional characteristics* such as hatred and desire for revenge. These intangible characteristics, as Allee (Allee 2002) mentioned, "are open to the flow of energy and matter. They exist on the edge of chaos. With too much openness, they disintegrate; with too little they become rigid and closed". These statements indicate the difficulty of representing intentional characteristics.

System dynamics modeling is limited in modeling intentional characteristics in a social-technical system. This is an inherited limitation because system dynamics modeling was originally proposed to simulate physical and mechanical systems, where intentional characteristics are not explicitly identified and modeled. Furthermore, general systems theory (Weinberg 2001), which includes system dynamics as a branch, intends to represent a system in terms of physical objects, and therefore does not fit for modeling intentions.

To a certain extent, intentional characteristics can be modeled by stocks, flows, and variables. For example, in Figure 1, we modeled `Fear of Punishment` and `Confidence in Water Quality` as stocks, and `(Water Operations) Fear of Punishment` as a variable. However, it becomes complicated when trying to analyze the system evolution and adaptation associated with these intentional stocks and variables. First of all, it is difficult to carry out time path analysis because the evolution of intentional characteristics does not follow any regular pattern. It is a direct mapping from a series of numbers to the plot of a time path, but it is not practical to give similar quantitative measures to intentions. How could the level of `Confidence in Water Quality` of a water operator be measured? This measure may be approximated in a

posterior manner, where the water operator recalls his historical confidence levels; but even when there is a satisfying approximation, the water operator's historical confidence levels cannot be used to model his current or future confidence levels because human cognitive model changes subtly yet significantly over the time; furthermore, one water operator's pattern of confidence levels (if such a pattern can be built) cannot be applied to model the confidence levels of other water operators because people think and behave in different ways.

In addition, intentional characteristics are special system features in that they are meaningful only when associated with the appropriate social actor (i.e., the owner of the intention); but due to the ambiguous subject and ambiguous object problems, it is difficult to identify the owners of intentional characteristics even when they are embedded in the system dynamics model as stocks, flows, and variables. This inherited limitation cannot be resolved unless the ambiguous subject and ambiguous object problems are solved.

2.2.3 Limitation on Modeling Autonomy

System dynamics modeling does not identify the *autonomy* of social components, which has three levels of meaning.

First, a social component may decide not to fulfill a task which is necessary for itself or other social components to achieve a goal. For example, the provincial government may decide not to provide enough budget to MOH, making MOH unable to oversee water test lab and water operations effectively; MOH may decide to cancel water operator training due to budget cut, making it difficult for water operator to achieve certain level of professional competence. This information is not explicitly represented in the system dynamics model.

Second, a social component has the freedom of choosing its strategies for achieving its goals. For example, a water operator's competence can be improved by organizational training (which can be cancelled due to budget cut), by professional or college training (which depends on personal intention and budget), by self-study (which depends on the operator's intention), or by a combination of multiple approaches. The system dynamics model is not able to illustrate the freedom of the operator for choosing different strategies for improving his competence.

Third, a social component can have conflicting goals and intentional characteristics, and it is up to the social component to resolve the conflicts and take appropriate actions. For example, for provincial government, reducing red tape is a goal that can be achieved by cutting budget; and minimizing risk of drinking water infection is another goal that can be achieved by sufficient and effective water operator training and routine inspection of the drinking water quality control process, which have to be guaranteed by enough budget support; so reducing red tape and minimizing risk of drinking water infection have conflict of interests. As another example, the water operator's confidence in untreated water quality leads to his negligence in following the standard drinking water quality control procedure, which conflicts with his fear of punishment for not following the standard procedure. Conflicts of goals and intentional characteristics are impossible to represent in system dynamics models.

2.2.4 Limitation on Modeling Goals

System dynamics model does not represent the goal related to each stock or variable; and without knowing the goals, it becomes difficult to understand why a stock or variable is present in the model. For example, why is `Oversight` needed in Figure 1? What is the goal of `Compliance with Quality Control Process`? To some extent, the ultimate goal of the

drinking water quality control system can be associated with the stock `Risk of Drinking Water Infection`, which implicitly states that the ultimate goal is to protect public health from potential drinking water infection. Although there is such an implicit meaning, the ultimate goal is not directly reflected in the model; in addition, it is not clear which social component is responsible for achieving this goal. In practice, this goal should be assigned to the provincial government, and then delegated to lower level governmental units and finally to the water operation management team and the water operators. This delegation of goals is obviously unavailable in the system dynamics model.

2.2.5 Limitations on Graphical Analysis

System dynamics model embeds adaptive system characteristics into the graphical model, but it does not provide graphical reasoning power for presenting the analysis process of system evolution and adaptation. As a result, the semantics of the adaptive characteristics have to be described by the analyst (usually in plain texts, like those in (Leveson, Daouk, Dulas, and Marais 2004)).

However, system dynamics model has the potential of performing semi-automated graphical analysis powered by the cause-effect links with polarity signs. The analysis starts by assigning system evolution values (increasing or decreasing) to an entity in the model, it then follows cause-effect links to collect meaningful system adaptation scenarios, and finally the effect on each scenario is evaluated by grouping the effects of all the links on the scenario. Furthermore, graphical components (e.g., starting and ending entities of a scenario, increasing and decreasing effects of entities in a scenario) can be invented to graphically express this analysis process in a system dynamics model. Note that rapid increase on the complexity of the model can lead to dramatic growth of the amount of information to be consumed during the analysis, which can make the described analysis process less practical. An in-depth study of methods in this area, especially searching algorithms for directed graphs, is required before proposing any feasible approach.

Based on the discussions about the limitations of system dynamics modeling, it follows that although system dynamics provides powerful support for modeling system evolution and adaptation, it is not enough to provide thorough analysis and modeling due to several inherited limitations. In the next section, we will explore the capability of the *i** intentional modeling framework for representing and reasoning adaptive requirements.

3 *i** Framework

3.1 Overview

*i** is a modeling framework extensively used in the early phase research of requirements engineering. The three fundamental concepts in *i** are actors, intentional elements, and links. An *actor* is an entity in the problem domain that takes strategic actions in order to achieve its goals. An actor can be further refined to be an *agent* (e.g., a person or a software system) or a *role* (e.g., a doctor or a manager). *i** has several *intentional elements*. A *goal* represents either a concrete object or a condition that an actor wants to obtain or achieve. A *softgoal* is a quality requirement whose satisfaction can not be accurately and quantitatively evaluated (e.g., efficiency and usability). A *task* represents an action that an actor performs to satisfy a goal. A *resource* represents a physical or informational entity whose existence is necessary for carrying out a task. *Intentional links* represent relationships between various actors, goals, softgoals, tasks, and resources. Common intentional links include *dependency*, *means-ends*, *decomposition*, and *contribution*. There are two types of

models in *i**: a *strategic dependency* (SD) model represents the dependency relationships between different actors in the problem domain [1], and a *strategic rationale* (SR) model depicts the internal configurations of an actor's intentional elements in order to achieve the actor's goals. Detailed explanation of *i** concepts can be found in (Yu 1995; Yu 1997).

Since its introduction, the *i** framework has been used by a number of researchers for conducting requirements analysis, modeling business processes and organizational structures, and analyzing privacy, security, and trust issues in social-technical environments (Kethers, Gans, Schmitz, and Sier 2005; Liu, Yu, and Mylopoulos 2002; Liu, Yu, and Mylopoulos 2003; Yu and Cysneiros 2002; Yu, Lin, and Mylopoulos 2007). All the previous research not only demonstrated the potential power of *i** in modeling socio-technical systems and environments, but also enriched the building blocks of the *i** framework for representing subtle issues (e.g., privacy, trust, belief, and security) in our society. Compared with behavioral models, *i** intentional models are considered to be more natural and appropriate for representing modern information systems where technologies and human social activities are closely intertwined.

In this section, we attempt to investigate the capability of *i** models for representing and reasoning adaptive requirements in social-technical systems. We carry out this investigation by applying SD modeling and SR modeling on the same drinking water quality control problem domain as discussed in the previous section.

3.2 SD Model

Figure 2 shows the SD model of the drinking water quality control system. We analyze this model in terms of the limitations of system dynamics model explained in the previous section, and we carry out our analysis by discussing the advantages and limitations of the SD model with respect to those of system dynamics.

3.2.1 Modeling Subject and Object

Advantages: The ambiguous subject and ambiguous object problems do not exist in SD model. Each system dynamics stock, flow, or variable, if it is identifiable in the SD model, is directly associated with a dependency relationship in the SD model. A dependency relationship contains a *depender*, a *dependee*, and a *dependum*. No matter whether the stock, flow, or variable is mapped to the depender, dependee, or dependum, there is always a subject and an object associated with the entity. For example, variable `Budget` in the system dynamics model does not give hint on who is responsible for providing budget to whom; and this information is still unclear by following all the links to and from the `Budget` variable. However, the SD model clearly demonstrates that `Provincial Government` is responsible for providing `Budget` to `MOH`, who is then responsible for providing `Budget` to `Water Operations`. As another example, it is not clear to know the subject and object of variable `Oversight` in Figure 1, but it is explicit to identify the `Oversight` relationship between `MOH` and `Water Test Lab` as well as that between `MOH` and `Water Operations` in Figure 2. Furthermore, the scattered responsibilities in the system dynamics model are now coherently assigned to specific actors in the SD model.

Limitations: Some of the entities in the system dynamics model are not representable in the SD model. This situation is especially true for some intentional characteristics, and it happens because the subject and object of the entity turn out to be the same social component. For example,

[1] SD is often used to denote system dynamics in the literature. In this paper, we always use the full name of system dynamics, and SD is used to refer to *i** strategic dependency model.

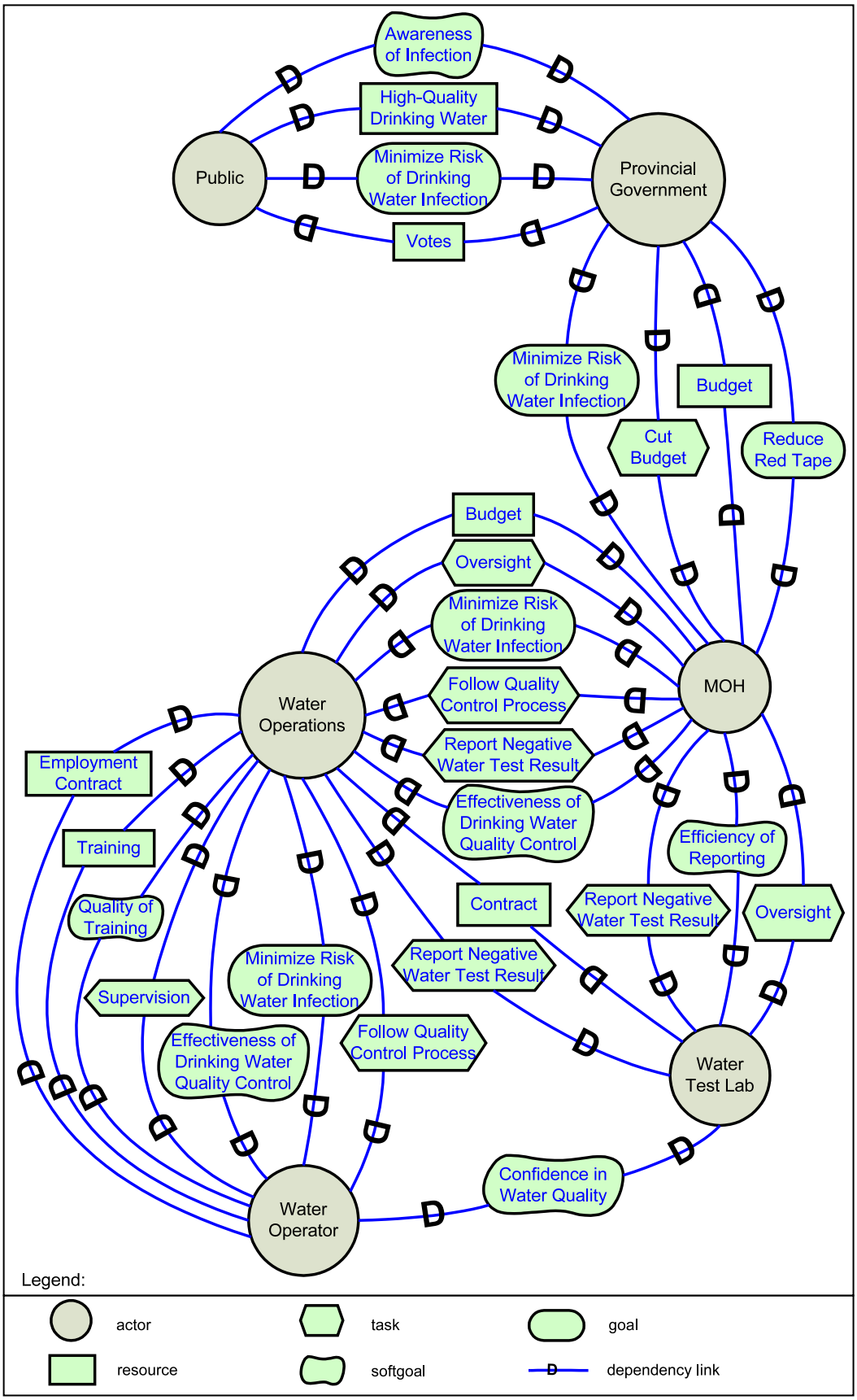

Figure 2. SD Model of Drinking Water Quality Control System

the Fear of Punishment of Water Operator is an intention dependent on the Water Operator himself; therefore it is not represented in the SD model due to the incapability of SD model for expressing unary dependency relationships.

3.2.2 Modeling Intentions

Advantages: Stocks, flows, and variables associated with intentional characteristics in the system dynamics model are modeled in SD such that some intentional characteristics are assigned to the corresponding actors who own those intentions. For example, variable Confidence in Water Quality corresponds to a softgoal dependency Confidence in Water Quality between Water Operator and Water Test Lab, which means that Water Operator has a level of confidence in the water quality, and this confidence level is affected by the water test results from Water Test Lab.

Limitations: Note that intentional characteristics are not thoroughly represented in the SD model. On the one hand, the dependency relationships used to model an intention may not be the sole determining factor of the intention. For example, though water test result from Water Test Lab can affect Confidence in Water Quality of Water Operator, there are a list of other factors that can determine the confidence level, such as the water operator's work experience, drinking habit, trust tendency, and peer pressure. On the other hand, it may not be appropriate to model some intentional characteristics as dependency relationships. For example, it is not suitable to represent Fear of Punishment of Water Operator as being dependent on other social components (i.e., Water Operations); instead, Fear of Punishment is more closely associated with Water Operator, the owning actor of the intention.

To summarize, the SD model is able to represent some but not all intentional characteristics in a social-technical system. We will soon explain in the next subsection that those intentional characteristics not caught in the SD model can be represented in the SR model.

3.2.3 Modeling Goals and Autonomy

Advantages: Autonomy indicates the degree of freedom of a social component when dealing with activities in a social-technical system. The SD model is agent-oriented in that it starts by identifying all the social components in a system. The SD model is also goal-oriented, meaning it associates each social component with its goals; and the responsibility of each goal is related with other social components. Based on these design principles, the SD model is good at modeling goals and autonomy in general (we discuss this generality in details below).

Limitations: The SD model is able to express the first level of meaning of autonomy, which is the freedom of a social component for not fulfilling a dependency responsibility. However, the SD model is not strong for reasoning the consequence of this dependency nondelivery. Taking the dependency relationships among Provincial Government, MOH, and Water Operations in Figure 2 as an example, Figure 4 (see Appendix A) [2] illustrates possible consequences when Provincial Government intends to Reduce Red Tape by cutting Budget. This figure does not follow any modeling standard but is similar to UML sequence diagram, and its objective is to demonstrate the sequential effects of one changing entity on other entities. Figure 4 illustrates that failing to deliver enough Budget to MOH may compromise public safety goal of Provin-

[2] This figure, as well as several other figures, are attached in a supporting appendix file, which is separate from the main paper due to file size limit.

cial Government to Minimize Risk of Drinking Water Infection, though it can be an effective way for achieving another goal to Reduce Red Tape. This kind of analysis is desirable but not available in the SD model.

It is worth noting that the sequential consequence analysis depicted in Figure 4 is available in system dynamics model, where the cause-effect links with polarity signs illustrate potential system evolutions and their effects. This is in fact the most significant limitation of SD modeling as compared with system dynamics modeling.

In addition, the SD model is not strong at expressing the second and third levels of autonomy. SD model is a high-level depiction of the dependency relationships in a system, so it does not pay attention to the ways of fulfilling the goals of each actor; neither does it model the relationships among various dependency relationships (and this relationship of relationships may be conflicting). As a result, the SD model is not able to show the autonomy of a social component for choosing different strategies to achieve its goals; neither is it able to express and resolve the conflicts between different goals, softgoals, and intentional characteristics.

3.2.4 Graphical Analysis

The capability of performing graphical analysis in SD model is even weaker than that in system dynamics model. The signed cause-effect links in a system dynamics model give the analyst a starting point of carrying out semi-automated graphical analysis. As described in section 2.2.5, the analyst can assign evolution values to selected entities in the model and collect meaningful system adaptation scenarios by tracing cause-effect links. The SD model does not even have the potential for conducting similar graphical analysis.

Improvement on graphical analysis power of SD model has been hinted in section 3.2.3 (Figure 4). However, the graphical analysis in Figure 4 seems to deviate too much from the current design of *i**. In other words, adding this graphical analysis feature requires extensions of *i** that are not cognitively straightforward and the potential benefit does not convincingly surpass the added complexity. We will come back to this topic later in section 4.

3.3 SR Model

In an SR model, intentional relationships are modeled internally by means-ends, task-decomposition, and contribution links; and externally by connecting proper internal intentional element to external actors through dependency links. Individual SR models can be constructed for each actor in the problem domain to explain the requirements from the viewpoints of all actors in the SD model. A complete SR model is then built by combining individual SR models. In this section, we investigate the capability of SR model from the viewpoint of MOH in the drinking water quality control system. This SR model is presented in Figure 5 (see Appendix A). For SR models from the viewpoints of other actors and the complete SR model, readers can refer to (Du 2006). Since SR model is based on SD model, it retains all the advantages of SD model. In fact, a complete SR model of a system contains the SD model of the system implicitly. We therefore discuss the advantages and limitations of SR model in terms of the limitations of SD model explained in the previous subsection.

3.3.1 Modeling Subject, Object, and Intention

We mentioned that some of the entities, especially intentional characteristics, are not representable in the SD model because the subject and the object of the entity are the same social component,

and SD model cannot express unary dependency relationships. SR model solves this problem by including all the associated entities within its actor boundary. For example, `Fear of Punishment` and `Confidence in Water Quality` can both be modeled as softgoals of `Water Operator` in the SR model. In Figure 5, `Trust in Water Test Lab` and `Trust in Water Operations` are modeled as softgoals of `MOH`, and their influence on other entities within the actor boundary is justified using contribution links.

SR model with actor boundary is a proper way of representing intentional characteristics because the ownership of each intention is clearly stated. In addition, the controlling entities of an intention as well as the entities controlled by the intention are easily expressible in SR model. For example, the ‘‘`Hurt`’’ contribution links from `Trust in Water Operations` to `Effectiveness and Adequacy of Inspection` and `Efficiency and Effectiveness of Advisories` imply that overtrust on water operations can compromise the achievement of MOH's top goal `Minimize Risk of Drinking Water Infection`. We can further model `Trust in Water Operations` to be dependent on the historical performance of water operations (which is not shown in Figure 5).

3.3.2 Modeling Autonomy

The SR model is able to model all the three levels of autonomy. First, the freedom of not fulfilling a dependency responsibility is represented since all the other actors and the associated dependency relationships are included in the SR model. Second, by using means-end and decomposition links, SR model can express the autonomy of a social component for choosing different strategies to achieve a goal or a task. For example, `MOH` can `Post on Radio`, `Post on Newspaper`, or `Post on TV` in order to `Post Drinking Water Advisories`. Third, conflicts between different goals and softgoals can be represented using contribution links between them. For example, the ‘‘`Hurt`’’ contribution links from sub-tasks of `Cut Budget` to sub-entities of `Minimize Risk of Drinking Water Infection` show that cutting budget can conflict with MOH's goal of minimizing risk of drinking water infection.

However, the SR model is still not good at reasoning the consequence of nondelivered dependency responsibilities. In other words, the effects shown in Figure 4, which are expressible in system dynamics model but not in SD model, are not representable in SR model. The conclusion is that *i** is not capable of modeling and reasoning sequential (or time-based) system evolution and adaptation.

3.3.3 Graphical Analysis

Graphical analysis in SR model is stronger than that in SD model and comparable to that in system dynamics model. The intentional links in SR model make it possible to perform graphical reasoning of cause-effect relationships similar to that conducted on system dynamics model. As indicated in Figure 6 (see Appendix A), the meaning of intentional links in SR model is similar to that of cause-effect links in system dynamics model. Two suggestions follow: 1) it is possible to carry out similar cause-effect analysis using existing intentional links in SR; and 2) it is possible to integrate one method into the other method if necessary.

Graphical analysis in SR is also supported by the qualitative evaluation method, which is useful for answering decision making questions. An evaluation scenario of the SR model starts by labeling all leaf intentional elements with known domain knowledge. The labels then propagate to their higher level elements following the *i** labeling rules. The root goal of the actor is finally labeled

as either satisfied or denied with a level of severity, suggesting whether the initial configurations of the system is subject to safety hazard and needs modification. Readers can refer to (Du 2006) for a qualitative evaluation example in the drinking water quality control domain.

3.4 Summary of SD, SR, and System Dynamics Models

We have discussed the advantages as well as limitations of system dynamics model, SD model, and SR model. On the one hand, system dynamics model is clear at identifying the dynamic evolution and adaptation of a social-technical system, which can be crucial in performing time-based reasoning and simulation, as well as analyzing the causes of an accident. On the other hand, *i** (SD and SR) models have the benefit of clearly representing the goals of social components and assigning the responsibility of each goal to the right actor(s); furthermore, *i** models are good at representing intentional characteristics and autonomy of social components. Since both system dynamics modeling and *i** modeling have their own advantages, it can be beneficial to reuse the desirable features of system dynamics modeling in *i** modeling, and vice versa. It may even be possible to combine these two modeling techniques together to form a new modeling approach.

4 Mapping System Dynamics and *i** Model Elements

The previous two sections discussed the advantages and limitations of system dynamics and *i** modeling separately. In this section, we attempt to put these two types of models side by side in order to give a direct comparison between their modeling features. We do this by performing a mapping between system dynamics model elements and *i** model elements. This mapping consists of two parts: mapping model elements between system dynamics and *i** SD, and mapping model elements between system dynamics and *i** SR. The example system dynamics model is the one given in Figure 1; the example SD model is the one given in Figure 2; and the example SR model is the one shown in Figure 5. In the following two subsections, we first discuss the mapping between the system dynamics model and the SD model, we then explain the mapping between the system dynamics model and the SR model.

4.1 Mapping Between System Dynamics and SD

Figure 7 (see Appendix A) shows the mapping between system dynamics model elements and *i** SD model elements. The left part of Figure 7 is a system dynamics model of the drinking water quality control system, and the right part is an SD model created for the same problem domain. The mapping process is bidirectional: for each system dynamics element, find its counterpart in the SD model; and for each SD element, find its counterpart in the system dynamics model. We found it easier for us to manipulate this mapping process if we start by identifying the subject and object actors in the SD model for each element in the system dynamics model [3]. The mapping process we used is summarized by the algorithm in Table 1, which is justified as follows using an example mapping illustrated in Figure 8 (see Appendix A).

First, for each system dynamics element (stock, flow, and variable) e and each outgoing causal link l connecting e to another element f, one or more subject-verb-object sentences are built to describe the relationship between the two elements connected by the causal link. The context of a system dynamics element e, including e itself, the causal link l, and the destination element f, is required to build these sentences. For example, in Figure 8, variable `MOH Oversight`, together

[3] This preference may be due to our experience with SD modeling, and it shall not be a general guideline for carrying out the mapping process; other users may find it more appropriate to conduct the mapping in other orders

with four outgoing causal links and their destination elements, leads to the construction of eleven sentences.

Second, every sentence is used to identify the depender, dependee, and dependum associated with the relationship described by the sentence. For example, in Figure 8, sentence a1 corresponds to the `Oversight` task dependency from depender `Water Operations` to dependee `MOH`, and sentence c2 refers to the `Quality of Training` softgoal dependency from actor `Water Operator` to actor `MOH`.

Third, we explicitly map the identified intentional element (i.e., the dependum) to either the source element e or the destination element f mentioned at the beginning of the mapping process. In Figure 8, these mappings are shown by lines connecting the five system dynamics variables and the nine *i** intentional elements.

Table 1. System Dynamics and *i SD Model Elements Mapping Process**

S: system dynamics model D: *i** SD model	
1	**For** each element e in S, **Do**
2	**For** each link l from e to another element f, **Do**
3	Build a set of subject-verb-object sentences L from e, l, and f
4	**For** each sentence t in L, **Do**
5	Find the intentional element i in D associated with t
6	Map i to either e or f
7	**End Do**
8	**End Do**
9	**End Do**

The mapping of actors in the SD model to elements in the system dynamics model is not explicitly shown, but it is implied during the mapping process by identifying subject and object actors of the each system dynamics element. This implicit process gives the effect that each system dynamics element is assigned an owning actor who is responsible for the delivery of that element. For example, variable `MOH Oversight` belongs to actor `MOH`, while actor `Water Operations` is responsible for variable `Effectiveness of Equipment Maintenance`.

4.2 Mapping Between System Dynamics and SR

Figure 9 (see Appendix A) shows the mapping between system dynamics model elements and *i** SR model elements. The left part of the figure is the same system dynamics model of the drinking water quality control system, and the right part is the SR model from the perspective of `MOH`. To map system dynamics elements with SR elements, we first need to go through the process in section 4.1 to obtain the mapping between system dynamics and SD models. This preliminary step is needed because the element decomposition inside an SR model is based on the dependency relationships of the SD model. The mapping process is summarized by the algorithm in Table 2.

The system dynamics and SR mapping process consists three parts. The first part (i.e., line 1) generates the mapping between the system dynamics model and the SD model using the algorithm in Table 1. The second part, including line 2 through line 5, transforms the mapping created in part one to SR model. This is arranged because the SD model is embedded in the SR model. The

Table 2. System Dynamics and i^* SR Model Elements Mapping Process

S: system dynamics model D: i^* SD model R: i^* SR model from perspective of actor A	
1	Map S and D
2	**For** each mapping between e_s in S and e_d in D, **Do**
3	Find the corresponding element e_r in R
4	Map e_s and e_r
5	**End Do**
6	**For** each element e_r in R that is mapped, **Do**
7	**For** each element e in the decomposition structure of e_r, **Do**
8	Find all the corresponding elements E in S
9	Map e to all elements in E
10	**End Do**
11	**End Do**

third part includes line 6 to line 11, and its purpose is to iterate through the element decomposition structure inside the SR model and map each encountered intentional element to its potential counterparts in the system dynamics model.

4.3 Suggestions of the Mapping Process

The main purpose of mapping system dynamics and *i** elements is not to show the mapping results themselves, but to reveal hints that can motivate the integration of system dynamics modeling features (i.e., time-based evolution and adaptation analysis) into *i** models. Now that we have demonstrated the mapping processes between system dynamics model and *i** SD/SR models, we can give a summary of findings from this mapping experience.

4.3.1 Cardinality of Mapping

The mapping process shows that each system dynamics element can be mapped to 0, 1, or more *i** elements, and vice versa. This relationship is demonstrated by the diagram in Figure 11 (see Appendix A). The following examples are found in Figure 7:

- 1-to-0: Variable `Fear of Punishment` maps to no *i** element.
- 1-to-1: Stock `Confidence in Water Quality` maps to softgoal `Confidence in Water Quality`.
- 1-to-*: Variable `MOH Oversight` maps to three *i** elements.
- 0-to-1: No variable/stock maps to resource `Votes`.
- *-to-*: One variable and one stock map to several goals `Minimize Risk of Drinking Water Infection`.

It is quite common to identify one-to-many cardinality between system dynamics and *i** elements, and this is primarily caused by the common existence of *delegation of responsibility* in *i** models, which means the assignment of responsibility from one actor to another actor. For example, stock `Risk of Drinking Water Infection` is first mapped to goal `Minimize`

Risk of Drinking Water Infection owned by Provincial Government; and it is then delegated to MOH, who further delegates the responsibility to Water Operations. Delegation of responsibility is a pervasive phenomenon in social-technical systems. We can see that it is difficult to use system dynamics model to demonstrate this feature.

4.3.2 Migration of Causal Links

Another suggestion of the mapping process, as one step further from the delegation of responsibility feature, is the potential migration of causal links to *i** models. To explain this finding more clearly, let us revisit the causal link from variable MOH Oversight to variable Compliance with Quality Control Process in Figure 8. This part of mapping is re-generated in Figure 10 (see Appendix A). The migration of the causal link from system dynamics model to SD model is justified as follows:

1. From sentence a1, source variable MOH Oversight maps to task Oversight.
2. From sentence a2, destination variable Compliance with Quality Control Process maps to task Follow Quality Control Process and softgoal Effectiveness of Drinking Water Quality Control.
3. Therefore, the causal link from the source variable to the destination variable is migrated to the SD model from task Oversight to task Follow Quality Control Process, as well as from task Oversight to softgoal Effectiveness of Drinking Water Quality Control.
4. From sentence a3, source variable MOH Oversight maps to task Supervision.
5. From sentence a4, destination variable Compliance with Quality Control Process maps to task Follow Quality Control Process and softgoal Effectiveness of Drinking Water Quality Control.
6. Therefore, the causal link from the source variable to the destination variable is migrated to the SD model from task Supervision to task Follow Quality Control Process, as well as from task Supervision to softgoal Effectiveness of Drinking Water Quality Control.
7. Based on delegation of responsibility, add one causal link from Oversight to Supervision, one between the two Effectiveness of Drinking Water Quality Control softgoals (from left to right), and one between the two Follow Quality Control Process tasks (from left to right).

Triggered by one causal link in the system dynamics model, seven causal links are added to the SD model in total. Adding these causal links to the SD model makes it possible to analyze the evolution relationships between intentional elements, such as that increase of Supervision can lead to improvement of task Follow Quality Control Process. However, keep in mind that while causality denotes a definite evolution relationship between two elements, the actual evolution relationship between two intentional elements is intentional instead of causal. Intentionality implies actor autonomy, meaning that it is up to the owning actor's discretion to determine whether or not to deliver a responsibility. Therefore increase of Supervision on Water Operator may not cause Water Operator to Follow Quality Control Process more closely. This intentional instead of causal characteristic is commonly available in social-technical systems. It is thus not acceptable to directly migrate causal links to *i** models. In the next section, we

will propose the integration of evolution-adaptation analysis into SD models with consideration of intentionality.

5 Towards a Goal-oriented Adaptive Requirements Engineering Approach

In this section, we describe a preliminary proposal of *GARE*, a goal-oriented adaptive requirements engineering approach for social-technical systems. The *GARE* approach is based on the discussions of the advantages and limitations of system dynamics modeling and *i** modeling in previous sections, and it is obtained by integrating time-based evolution and adaptation analysis into *i** models. Our current proposal focuses on extending SD models only. Adding evolution and adaptation analysis to SR models is one of our future works.

5.1 Evolution Labels

In order to model time-based evolution of intentional elements, we can associate time paths with intentional elements in SD models so that system evolution over the time can be described. While a time path quantitatively describes the evolution of an element, this level of quantitative knowledge is unavailable in many situations, where qualitative representations fit more appropriately. In this paper, an *evolution label* $L(e)$ attached to an intentional element e is used to qualitatively describe the time-based evolution of e. Eight types of evolution label are introduced in this paper:

- $L_+(e)$: linear-like increasing. The level of element e increases at a constant rate (over time).
- $L_-(e)$: linear-like decreasing. The level of element e decreases at a constant rate.
- $E_+(e)$: exponential-like increasing. The level of element e increases at a rate that is increasing.
- $E_-(e)$: exponential-like decreasing. The level of element e decreases at a rate that is increasing.
- $G_+(e)$: logarithmic-like increasing. The level of element e increases at a rate that is decreasing.
- $G_-(e)$: logarithmic-like decreasing. The level of element e decreases at a rate that is decreasing.
- $S(e)$: stable state without change. The level of element e does not change.
- $N(e)$: unknown evolution. The evolution of the level of element e is unknown.

By using the "*-like*" suffix, these labels are associated with the *qualitative property*, meaning that, instead of referring to a specific mathematical function, each label is used to describe a class of evolutions like the one denoted by its identifier; and the boundary of this class is not clearly specified. For example, label $G_-(e)$ indicates that the level of e first decreases greatly, and then continues to decrease but at a lower rate, similar to a logarithmic-style decreasing; however, the actual plot of this decreasing curve, whether it is a logarithmic function or a reciprocal relation, is not a concern in the discussion. The idea of this qualitative representation is taken from qualitative reasoning (Forbus 1988; Kuipers 1994), where landmark values and fuzzy sets are used to qualitatively represent real values of objects. Graphical notations of evolution labels are illustrated in Figure 3.

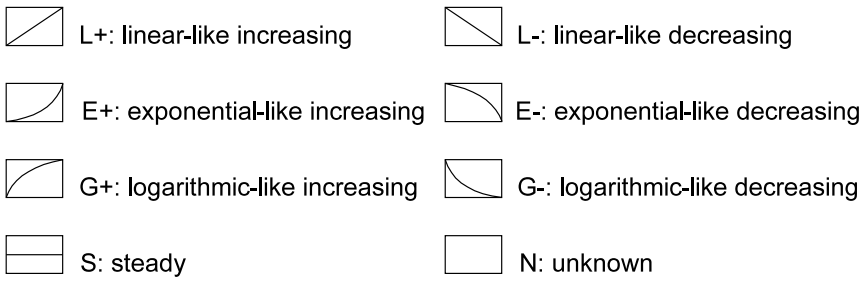

Figure 3. Evolution and Adaptation Labels

5.2 Adaptation Labels

The fundamental characteristic of social-technical systems is that the relationships between system components are dynamic and affecting each other. Evolution labels describe how intentional elements change over the time, it then becomes important to understand how the evolution of one element affects the changes of other elements.

An *adaptation label* $L(s, d)$ is used to qualitatively describe how intentional element d changes with the evolution of intentional element s. Note that an evolution label describes the relationship between an intentional element and the time, while an adaptation label represents the relationship between two intentional elements. Similar to evolution labels, we discuss eight types of adaptation labels in this paper, and the relationships described by them are comparable to those of evolution labels:

- $L_+(s, d)$: linear-like increasing.
- $L_-(s, d)$: linear-like decreasing.
- $E_+(s, d)$: exponential-like increasing.
- $E_-(s, d)$: exponential-like decreasing.
- $G_+(s, d)$: logarithmic-like increasing.
- $G_-(s, d)$: logarithmic-like decreasing.
- $S(s, d)$: stable state without adaptation.
- $N(s, d)$: unknown evolution.

As shown in Figure 3, adaptation labels share the same set of graphical notations with evolution labels.

5.3 Adaptation Links

Adaptation labels need to be attached to the SD model before any evolution-adaptation analysis can take place. Adaptation links are the only places in an SD model where adaptation labels can be associated with.

An *adaptation link* $A(a, L(s, d), i)$ is a directional link within the boundary of actor a, labeled with adaptation label L from source intentional element s to destination intentional element d, and with intentionality i. Adaptation links are adapted from causal links based on the discussion of possibility for migrating causal links to *i** models in section 4.3.2, where we mentioned the limitations of a direct migration (e.g., lack of intentionality). We now justify two important properties of adaptation links as follows:

- **actor-oriented**: each adaptation link belongs to an actor who controls its construction, modification, destruction, and level of intentionality.
- **intentionality**: while a causal link in system dynamics indicates an inevitable cause-effect relationship between two elements (i.e., change of the source element leads to change of the destination element definitely), the relationship denoted by an adaptation link between two intentional elements is intentional instead of causal, which means the evolution of the source element may or may not lead to the change of the destination element. The level of intentionality, i, is used to indicate how likely an adaptation link is going to take effect. We discuss the intentionality of adaptation links in Step 5c of section 4.6.

Figure 12 (see Appendix A) illustrates an example where adaptation labels are attached to four adaptation links within actor `MOH`. An adaptation link states how the destination intentional element changes with the evolution of the source intentional element. For example, the adaptation link from `Reduce Red Tape` to `Cut Budget` means the performance level of task `Cut Budget` changes linear-like-ly with the satisfaction level of goal `Reduce Red Tape`.

An actor in the SD model can serve as two roles: a depender with outgoing dependency links to intentional elements, or a dependee with incoming dependency links from intentional elements. In theory, an adaptation link can be added between any two intentional elements connected to an actor. If we categorize the source and destination elements of an adaptation link by the role the participating actor acts on, we can have four configurations. Let R be the set of dependency elements linking from actor α (i.e., α acts as the depender), and E the set of dependency elements linking to α (i.e., α acts as the dependee), then the source and destination elements of an adaptation link can have four combinations:

		destination element	
		$e_j \in E$	$r_n \in R$
source element	$e_i \in E$	1. valid	3. invalid
	$r_m \in R$	2. valid	4. invalid

Note that the first two configurations are valid while the other two are invalid. We justify them as follows:

1. $e_i \rightarrow e_j$, where $e_i \in E, e_j \in E$, and $i \neq j$: valid, meaning that the the ability of the participating actor α to fulfill responsibility e_j (as a dependee) changes with the ability of α to fulfill responsibility e_i (as a dependee).
2. $r_m \rightarrow e_j$, where $r_m \in R$ and $e_j \in E$: valid, meaning that the ability of the participating actor α to fulfill responsibility e_j (as a dependee) changes with the ability of another actor (as a dependee and α as a depender) to fulfill responsibility r_m.
3. $e_i \rightarrow r_n$, where $e_i \in E$ and $r_n \in R$: invalid, meaning that the ability of another actor β (as a dependee and the participating actor α as a depender) to fulfill responsibility r_n changes with the ability of α (as a dependee) to fulfill responsibility e_i. This type of adaptation link should be added to actor β where the second configuration applies.
4. $r_m \rightarrow r_n$, where $r_m \in R$, $r_n \in R$, and $m \neq n$: invalid, meaning that the ability of another actor β (as a dependee and the participating actor α as a depender) to fulfill responsibility r_n changes with the ability of yet another actor γ (as a dependee and α as a depender) to fulfill responsibility r_m. This type of adaptation link should be added to either actor β or actor γ where the first two configurations may apply.

We call an intentional element e a *depender element* of actor α if α acts as the depender of e. Similarly, e is a *dependee element* of α if α acts as the dependee of e. As a summary of the above configurations, the destination of an adaptation link must be a dependee element, and the source of an adaptation link can be either a dependee element or a depender element.

Figure 12 has the goal `Reduce Red Tape` attached with a linear-like increasing evolution label, and an adaptation link is added to `MOH` to connect this goal to task `Cut Budget`. It is now an intuitive request to derive the evolution label of task `Cut Budget`. This process is controlled by label propagation rules, which we present next.

5.4 Label Propagation Rules

Given two intentional elements s and d, if s is attached with an evolution label L_s and there is an adaptation link from s to d with adaptation label L_a, it is then desirable to predict the evolution label of the destination intentional element d. We use the notation $L_s \xrightarrow{L_a} L_d$ to represent the propagation process, indicating that source evolution label L_s and adaptation label L_a together derive destination evolution label L_d. The label propagation algorithm in Table 3 defines this process.

These label propagation rules are obtained by performing mathematical function transformations on the relations representing the source evolution label and the adaptation label. Specifically, let s and d be the source and destination intentional elements, then for the set of rules with source evolution label L_- (lines 8 and 9 of the algorithm in Table 3), the following transformations are carried out:

$$s = L_-(t) \begin{cases} \xrightarrow{d=L_+(s)} d = L_+(L_-(t)) = L_-(t) \\ \xrightarrow{d=L_-(s)} d = L_-(L_-(t)) = L_+(t) \\ \xrightarrow{d=G_+(s)} d = G_+(L_-(t)) = E_-(t) \\ \xrightarrow{d=G_-(s)} d = G_-(L_-(t)) = E_+(t) \\ \xrightarrow{d=E_+(s)} d = E_+(L_-(t)) = G_-(t) \\ \xrightarrow{d=E_-(s)} d = E_-(L_-(t)) = G_+(t) \end{cases}$$

Readers need to be reminded of the "*-like*" property of evolution and adaptation labels, meaning that the transformation is correct as long as the final evolution of the destination element qualitatively follows the shape specified by the label identifier. All the function transformations have been validated using the SciLab tool.

With label propagation rules specified, we are almost ready to perform a full labeling of an SD model. We need to solve two puzzles before gaining this full capability, and they are label aggregation and label distribution.

5.5 Label Aggregation and Distribution

Label aggregation is needed when an intentional element is labeled more than once via multiple adaptation links. For example, in Figure 13(A) (see Appendix A), resource `Votes` are labeled twice:

$L_+ \xrightarrow{E_+} E_+$, via route $a \xrightarrow{1} b$, and
$G_- \xrightarrow{L_+} G_-$, via route $c \xrightarrow{2} d$.

Table 3. Label Propagation Rules

Label Propagation Algorithm: given L_s and L_a, what is L_d?	
$A = \{L_+, L_-, E_+, E_-, G_+, G_-, S, N\}$	
$A_p = \{L_+, E_+, G_+\}$	
$A_n = \{L_-, E_-, G_-\}$	
L_s: evolution label of source element, $L_s \in A$	
L_a: label of adaptation link, $L_a \in A$	
L_d: evolution label of destination element, $L_d \in A$	
1	**If** $L_s \in \{S, N\}$, **then**
2	$\forall L_a \in A, L_d = L_s$
3	**Else if** $L_a \in \{S, N\}$, **then**
4	$\forall L_s \in A, L_d = L_a$
5	**Else if** $L_s = L_+$, **then**
6	$\forall L_a \in A, L_d = L_a$
7	**Else if** $L_s = L_-$, **then**
8	$L_s \xrightarrow{L_+} L_-$, $L_s \xrightarrow{G_+} E_-$, $L_s \xrightarrow{E_+} G_-$
9	$L_s \xrightarrow{L_-} L_+$, $L_s \xrightarrow{G_-} E_+$, $L_s \xrightarrow{E_-} G_+$
10	**Else if** $L_s = E_+$, **then**
11	$L_s \xrightarrow{L_+} E_+$, $L_s \xrightarrow{E_+} E_+$, $L_s \xrightarrow{G_+} L_+$
12	$L_s \xrightarrow{L_-} E_-$, $L_s \xrightarrow{G_-} L_-$, $L_s \xrightarrow{E_-} E_-$
13	**Else if** $L_s = E_-$, **then**
14	$L_s \xrightarrow{L_+} E_-$, $L_s \xrightarrow{G_+} E_-$, $L_s \xrightarrow{E_+} G_-$
15	$L_s \xrightarrow{L_-} E_+$, $L_s \xrightarrow{G_-} E_+$, $L_s \xrightarrow{E_-} G_+$
16	**Else if** $L_s = G_+$, **then**
17	$L_s \xrightarrow{L_+} G_+$, $L_s \xrightarrow{G_+} G_+$, $L_s \xrightarrow{E_+} L_+$
18	$L_s \xrightarrow{L_-} G_-$, $L_s \xrightarrow{G_-} G_-$, $L_s \xrightarrow{E_-} L_-$
19	**Else if** $L_s = G_-$, **then**
20	$L_s \xrightarrow{L_+} G_-$, $L_s \xrightarrow{G_+} N$, $L_s \xrightarrow{E_+} N$
21	$L_s \xrightarrow{L_-} G_+$, $L_s \xrightarrow{G_-} N$, $L_s \xrightarrow{E_-} N$
22	**End**

To know the overall evolution label of resource `Votes`, we must aggregate the two evolution labels obtained via the two routes. We use the notation $L_1 \oplus L_2$ to denote the label aggregation operation on two evolution labels L_1 and L_2. So in this example we need to calculate $E_+ \oplus G_-$. The label aggregation operation has two properties:

- **commutative**: $L_1 \oplus L_2 = L_2 \oplus L_1$, and
- **associative**: $L_1 \oplus L_2 \oplus L_3 = L_1 \oplus (L_2 \oplus L_3)$.

The algorithm in Table 4 defines label aggregation rules. With the commutative property, this algorithm only needs to specify rules with respect to all the cases of the first operant.

The label aggregation rules are derived from the big-O notation in the computational complexity theory of mathematical functions, where $O(1) = O(logx) = O(x) = O(e^x)$, indicating the

ascending order of the complexity of four mathematical functions used in our labeling set. Based on this ordering, we define two orderings of evolution labels: 1) $S < G_+ < L_+ < E_+ < N$, and 2) $S < G_- < L_- < E_- < N$.

If two evolution labels are in the same ordering, then their aggregation is the same as the label with higher order. Note the "unknown" evolution label has the highest order in both lists; and this is a reasonable setting because aggregating any evolution label with an unknown evolution label will result in an unknown evolution label. If two evolution labels are in different orderings, then their aggregation is always unknown (except for the S label, where the first rule applies); and this rule conforms to the qualitative property of evolution labels because the result of aggregating an increasing evolution label and a decreasing evolution label, without knowledge of their actual mathematical plots, is unknown.

Table 4. Label Aggregation Rules

Label Aggregation Algorithm: given L_1 and L_2, what is L_a?	
$A = \{L_+, L_-, E_+, E_-, G_+, G_-, S, N\}$	
$A_p = \{L_+, E_+, G_+\}$	
$A_n = \{L_-, E_-, G_-\}$	
L_1: evolution label, $L_1 \in A$	
L_2: evolution label, $L_2 \in A$	
L_a: aggregated evolution label, $L_a \in A$ and $L_a = L_1 \oplus L_2$	
1	**If** $L_1 = S$, **then**
2	$\forall L_2 \in A, L_a = L_2$
3	**Else if** $L_1 = N$, **then**
4	$\forall L_2 \in A, L_a = L_1$
5	**Else if** $L_1 \in A_p$ and $L_2 \in A_n$, **then**
6	$L_a = N$
7	**Else if** $L_1 \in A_p$ and $L_2 \in A_p$, **then**
8	$\forall L_2 \in A_p, E_+ \oplus L_2 = E_+$
9	$\forall L_2 \in \{L_+, G_+\}, L_+ \oplus L_2 = L_+$
10	$for L_2 = G_+, G_+ \oplus L_2 = G_+$
11	**Else if** $L_1 \in A_n$ and $L_2 \in A_n$, **then**
12	$\forall L_2 \in A_n, E_- \oplus L_2 = E_-$
13	$\forall L_2 \in \{L_-, G_-\}, L_- \oplus L_2 = L_-$
14	$for L_2 = G_-, G_- \oplus L_2 = G_-$
15	**End**

Label distribution happens when the evolution label of an intentional element is propagated to more than one intentional elements via multiple adaptation links. As illustrated in Figure 13(B), the label of resource `Budget` on the left are used twice: 1) $L_- \xrightarrow{L_+} L_-$, via route $e \xrightarrow{3} f$, and 2) $L_- \xrightarrow{G_+} G_-$, via route $e \xrightarrow{4} g$.

Label distribution can become complicated if we need to quantitatively measure the effect of a source evolution label on multiple adaptation links. In this paper, we only consider the qualitative characteristic of evolution labels, and we assume that a full copy of the source evolution label is distributed to every adaptation link which the source intentional element is connected with.

5.6 Analysis Process

In section 3, we have explained the incapability of SD models for performing time-based analysis. We now discuss a process for integrating evolution-adaptation analysis into SD models with the help of evolution labels, adaptation labels, adaptation links, and related label propagation and aggregation rules. The analysis process is shown in Figure 14 (see Appendix A). The following steps explain this process.

Step 1: Constructing Model

The analysis starts by constructing the initial SD model with available domain knowledge. SD model constructed at this stage contains no information about system evolution and adaptation. We use part of Figure 2 as our initial SD model. As shown in Figure 15 (see Appendix A), this model contains four actors and their dependency relationships.

Step 2: Adding Adaptation Links

Adaptation links are added to the initial SD model by studying the relationships between different dependency links from the viewpoint of every actor. Eleven adaptation links (indicated by numbers 1 to 11) are added to Figure 15. All these adaptation links are labeled based on prescriptive knowledge, meaning the relationships between the source and destination intentional elements are assumed. As an example, adaptation link 2 indicates that `Oversight` of `MOH` on `Water Operations` grows logarithmically with `Budget` from `Provincial Government`.

Step 3: Specifying Initial Evolution Labels

Initial evolution labels are assigned to selected dependency elements in order to trigger the label propagation process. In principle, every intentional element can be initially labeled, but some selections may not be practically meaningful. A general guideline is to follow the goal-oriented analysis nature of SD modeling by starting from the top goals of any actor. A case generator is responsible for configuring all possible initial labelings of the SD model. In Figure 15, `Budget` from `Provincial Government` to `MOH` is assigned the initial evolution label L_- (indicated by letter a), meaning the provincial government decides to linear-like-ly decrease budget to MOH. This initial labeling is motivated by the goal `Reduce Red Tape`. We need to propagate this initial evolution label on the whole SD model to see what system adaptations will take place accordingly.

Step 4: Propagating Evolution Labels

Figure 15 shows the result of first round label propagation, and the propagated evolution labels are indicated by letters b to j. Pay attention to labels d and h because they are obtained through label propagation followed by label aggregation. For example, label d is calculated by:

$$L_- \xrightarrow{L_+} L_-, \text{ via route } b \xrightarrow{4} d,$$
$$E_- \xrightarrow{G_+} E_-, \text{ via route } c \xrightarrow{5} d, \text{ and}$$
$$E_- \oplus L_- = E_-.$$

The labeling process in Figure 15 stops when label j is obtained on `Budget`, where the original evolution label a was specified. This process can continue by aggregating label j with the initial label a, and use the aggregated label for a second round propagation. The next step gives some analysis of the first round propagation.

Step 5: Analyzing System Adaptation

Step 5a: Feedback loop analysis: When the label propagation process finishes, conclusions on how the system adapts to the initial element evolution can be drawn by following feedback loops. *Feedback loop analysis*, which aims at finding out what effect an element e will receive if some initial evolution is applied to e, is one of the most significant features of system dynamics. Identifying and constructing feedback loops are important for building sustainable social-technical systems because well-constructed feedback loops help adapt a system back to equilibrium state, while badly-formed feedback loops tend to lead a system off balance.

The direct conclusion from the first round label propagation on Figure 15 is that linear-like-ly decreasing `Budget` from `Provincial Government` to `MOH` will cause unknown evolution of the `Budget` element itself. This conclusion is drawn from three feedback loops:

1. $a \xrightarrow{1} b \xrightarrow{4} d \xrightarrow{6} e \xrightarrow{3} f \xrightarrow{7} g \xrightarrow{10} h \xrightarrow{8} j$,
2. $a \xrightarrow{2} c \xrightarrow{5} d \xrightarrow{6} e \xrightarrow{3} f \xrightarrow{7} g \xrightarrow{10} h \xrightarrow{8} j$, and
3. $a \xrightarrow{9} i \xrightarrow{11} h \xrightarrow{8} j$.

Label propagation rules can be applied to each of these feedback loops to identify its adaptation effect on the system. Both the first and the second feedback loops turn out to propagate evolution label G_- on the initial intentional element, meaning they are positive feedback loops that will reinforce the budget decreasing policy of `Provincial Government`. The third feedback loop propagates evolution label E_+ on the initial intentional element, meaning it is a negative feedback loop that denies the initial budget decreasing policy.

The overall effects of these feedback loops on the initial intentional element, as mentioned just now, is unknown, which is obtained by label aggregation $G_- \oplus E_+ = N$. To draw more concrete conclusion about the overall system adaptation to the initial element evolution, more quantitative representations of evolution and adaptation labels are needed, which is not the topic of this paper.

Step 5b: Delay analysis: Delays can happen on both dependency links and adaptation links. Delay occurs on a dependency link if the dependee intentionally postpones the delivery of a responsibility. Delay takes place on an adaptation link if the owning actor intentionally detains or shuts off the evolution-adaptation relationship between the source and destination elements.

Delays can break the feedback effect of feedback loops. For example, if delay occurs on the `Oversight` dependency link between `MOH` and `Water Operations` (which means `MOH` intentionally delays the oversight on `Water Operations`), then the second feedback loop would break (indicated by the $\otimes$ sign):

- $a \xrightarrow{2} \bigotimes\ c \xrightarrow{5} d \xrightarrow{6} e \xrightarrow{3} f \xrightarrow{7} g \xrightarrow{10} h \xrightarrow{8} j$.

If delay also occurs on adaptation link 1 (caused by the unwillingness of `MOH` to allocate enough budget to `Water Operations`), then the first feedback loop would break too, resulting in leaving the third feedback loop to control the evolution-adaptation process. If, on the other hand, delay occurs on adaptation link 9 (caused by the intention of `Provincial Government` to abuse the cut budget), then the third feedback loop would break, which would leave the evolution-adaptation process to be controlled by the first feedback loop.

Step 5c: Intentionality analysis: In section 4.3, we defined that each adaptation link is associated with an *intentionality*, which means the evolution-adaptation effect between the source and destination elements may or may not take place, depending on the intention of the owning actor.

With intentionality added, the feedback analysis and delay analysis can no longer be taken for granted since actors along the feedback loops may decide to modify or destruct an adaptation link at any time.

There can be various approaches to enabling intentionality analysis. In this paper, we use a probabilistic method (Pearl 1988), in which the intentionality of each adaptation link is represented by a probability in the range $[0, 1]$. This intentionality level is determined, and may be modified, by the owning actor. A 0 intentionality means the the destruction of the adaptation link, while a 1 intentionality makes the adaptation link causal, indicating a definite adaptation effect. As an example, let us try to assign intentionality values to the adaptation links along the three feedback loops:

1. $a \xrightarrow[0.9]{1} b \xrightarrow[0.7]{4} d \xrightarrow[0.9]{6} e \xrightarrow[0.98]{3} f \xrightarrow[0.99]{7} g \xrightarrow[0.8]{10} h \xrightarrow[0.7]{8} j$,
2. $a \xrightarrow[0.8]{2} c \xrightarrow[0.9]{5} d \xrightarrow[0.9]{6} e \xrightarrow[0.98]{3} f \xrightarrow[0.99]{7} g \xrightarrow[0.8]{10} h \xrightarrow[0.7]{8} j$,
3. $a \xrightarrow[0.85]{9} i \xrightarrow[0.6]{11} h \xrightarrow[0.7]{8} j$.

With these prescribed intentionality levels, we can calculate the probability for a feedback loop to proceed at each intentional element along the loop. The calculations result in the overall probability for each feedback loop to complete the whole loop:

loop	yes	no
1	0.308	0.692
2	0.352	0.648
3	0.357	0.643

Based on the above calculations, we can finalize the analysis by computing the combined probability for the first two loops to finish and the third loop not to finish, which is obtained by: $(0.308 + 0.352) \times 0.643 = 0.424$. This analysis means that the probability for either feedback loop one or feedback loop two to take place and feedback loop three not to take place is 0.424. The conclusion drawn from this analysis is that with a 0.424 probability, the initial evolution will be reinforced, which may lead to unstable system state; and this is caused by feedback loops one and two. More complicated probabilistic analysis can be applied using Bayesian networks (Jensen 2001).

Step 6: Iterating Other Initial Labels

After the analysis of the current initial evolution label, the process returns to step three if there are other initial evolution labels (or the combination of multiple labels). A case generator is used to iterate through all possible initial labeling of the model. If there are too many cases, the case generator can apply some case generation strategy (Du and Hoffman 2004; Ramachandran 2003; Tai and Lei 2002) to decrease the number of iterations.

Step 7: Validating Analysis Results, and
Step 8: Updating Model

When all the cases are considered and correspondent analyses performed, the analysis results need to be validated before they can be applied to updating the initial model. The validation step is likely to be manual because it involves careful re-thinking of the whole environmental context and must be conducted by domain experts. Based on the survived suggestions after the validation step, the initial model can be updated to make the system adapt to element evolutions more appropriately.

6 Conclusion

This paper explored the capabilities of system dynamics modeling and *i** modeling for representing and reasoning adaptive requirements in social-technical systems. For either modeling approach, its advantages were explained and limitations summarized. We demonstrated that each modeling method has its advantages that are difficult to achieve using the other method, and each has its limitations that can be overcome using features available in the other method.

We proposed a goal-oriented adaptive requirements engineering (*GARE*) approach by integrating the *i** modeling framework with time-based evolution and adaptation analysis. *GARE* makes it possible to perform goal-oriented and agent-oriented adaptive requirements analysis using evolution lables, adaptation labels, and evolution-adaptation links. This approach is useful for modeling and analyzing adaptive requirements in both social-technical systems and other adaptive systems where social actors and their intentions motivate system changes, and where system evolution and adaptation are the fundamental characteristics.

This preliminary proposal of *GARE* acts as the first milestone of our attempt to engineering adaptive requirements. Future work will focus on four tasks. First, a complete integration of system dynamics features into *GARE* is needed, which includes a proposal of evolution-adaptation analysis on *i** SR models. Second, automated system simulation using *GARE* can be desirable to illustrate the effects of system evolution and adaptation. Third, empirical studies are needed to show whether a seamless integration of two methods is more powerful than using two methods separately. Fourth, formal representation of adaptive requirements is needed.

References

Abowd, G. D. and E. D. Mynatt (2000). Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction 7*(1), 29–58.

Allee, V. (2002). A value network approach for modeling and measuring intangibles.

Checkland, P. (1999). *Systems Thinking, Systems Practice*. John Wiley & Sons.

Cohen, D., S. Dalal, M. Fredman, and G. Patton (1997). The aetg system: An approach to testing based on combinatorial design. *IEEE Transactions on Software Engineering 23*(7), 437–444.

Du, Y. (2006, October). Towards adaptive requirements analysis and modeling. Research paper.

Du, Y. and D. Hoffman (2004). Pbit: A pattern-based testing framework for iptables. In *Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04)*, Fredericton, NB, Canada, pp. 107–112.

Finkelstein, A., A. Savigni, G. Kappel, W. Retschitzegger, W. Schwinger, and C. Feichtner (2002, jul). Ubiquitous web application development - a framework for understanding. In *Proceedings of the Sixth World Multi-Conference on Systemics, Cybernetics and Informatics (SCI2002)*, Orlando, FL, USA.

Forbus, K. (1988). *Qualitative Physics: Past, Present, and Future*, pp. 239–290. Morgan-Kaufmann.

Forrester, J. (1961). *Industrial Dynamics*. Cambridge, MA, USA: MIT Press.

Goldsby, H. and B. H. Cheng (2006, sep). Goal-oriented modeling of requirements engineering for dynamically adaptive systems. In *Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)*, Minneapolis, MN, USA, pp. 338–339.

Jameson, A. (2001). Modeling both the context and the user. *Personal and Ubiquitous Computing 5*(1), 29–33.

Jensen, F. V. (2001). *Bayesian Networks and Decision Graphs*. New York, USA: Springer-Verlag.

Kappel, G., B. Proll, W. Retschitzegger, W. Schwinger, and T. Hofer (2001). Modeling ubiquitous web application - a comparison of approaches. In *Proceedings of the 2001 International Conference on Information Integration and Web-based Applications and Services (iiWAS2001)*.

Kethers, S., G. Gans, D. Schmitz, and D. Sier (2005, may). Modelling trust relationships in a healthcare network: Experiences with the tcd framework. In *Proceedings of the 13th European Conference on Information Systems*, Regensburg, Germany.

Kobsa, A. (2001). Generic user modeling systems. *User Modeling and User-Adapted Interaction 11*(1-2), 49–63.

Kolos-Mazuryk, L., G.-J. Poulisse, and P. van Eck (2005). Requirements engineering for pervasive services. In *Proceedings of the OOPSLA Workshop on Building Software for Pervasive Computing*, San Diego, USA.

Kuipers, B. (1994). *Qualitative Reasoning - Modeling and Simulation with Incomplete Knowledge*. MIT Press.

Leplat, J. (1984). Occupational accident research and systems approach. *Journal of Occupational Accidents 6*, 77–89.

Leveson, N. (2004). A new accident model for engineering safer systems. *Safety Science 42*, 237–270.

Leveson, N., M. Daouk, N. Dulas, and K. Marais (2004, mar). A systems theoretic approach to safety engineering. In *Engineering Systems Monograph of the MIT Engineering Systems Symposium*, Cambridge, MA, USA.

Liu, L., E. Yu, and J. Mylopoulos (2002, oct). Analyzing security requirements as relationships among strategic actors. In *Proceedings of the 2nd Symposium on Requirements Engineering for Information Security (SREIS02)*, Raleigh, NC, USA.

Liu, L., E. Yu, and J. Mylopoulos (2003, sep). Security and privacy requirements analysis within a social setting. In *Proceedings of the 3rd IEEE Joint International Requirements Engineering Conference (RE'03)*, Monterey, CA, USA, pp. 151–161.

Niemela, E. and J. Latvakoski (2004). Survey of requirements and solutions of ubiquitous software. In *Proceedings of the 3rd International Conference on Mobile Ubiquitous Multimedia*, College Park, MD, USA, pp. 71–78.

O'Connor, D. R. (2002). Report of the walkerton inquiry, ontario ministry of the attorney general.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann.

Petak, W. J. (1981). Environmental management: A system approach. *Environmental Management 5*(3), 213–224.

Ramachandran, M. (2003, sep). Testing software components using boundary value analysis. In *Proceedings of the 29th Euromicro Conference*, Belek-Antalya, Turkey.

Randers, J. (1980). *Elements of the System Dynamics Method*. Cambridge, MA, USA: MIT Press.

Rasmussen, J. (1997). Risk management in a dynamic society: A modelling problem. *Safety Science 27*(2-3), 183–213.

Richardson, G. and A. Pugh (1981). *Introduction to System Dynamics Modeling with DYNAMO*. Cambridge, MA, USA: Productivity Press.

Roberts, N., D. F. Andersen, R. M. Deal, and W. A. Shaffer (1983). *Introduction to Computer Simulation: A System Dynamics Modeling Approach*. Reading, MA, USA: Addison Wesley.

Sterman, J. D. (1987). Expectation formation in behavioral simulation models. *Behavioral Science 32*, 190–211.

Sterman, J. D. (2000). *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Irwin/McGraw-Hill.

Sterman, J. D. (2002). All models are wrong: Reflections on becoming a systems scientist. *System Dynamics Review 18*(4), 501–531.

Tai, K.-C. and Y. Lei (2002). A test generation strategy for pairwise testing. *IEEE Transactions on Software Engineering 28*(1), 109–111.

Tsandilas, T. and M. Schraefel (2004). Usable adaptive hypermedia systems. *New Reviews of Hypermedia and Multimedia 10*(1), 5–29.

Vicente, K. and K. Christoffersen (2006). The walkerton e.coli outbreak: A test of rasmussen's framework for risk management in a dynamic society. *Theoretical Issues in Ergonomics Science 7*(2), 93–112.

Weinberg, G. M. (2001, April). *An Introduction to General Systems Thinking*. Dorset House.

Yu, E. (1995). *Modelling Strategic Relationships for Process Reengineering*. Ph. D. thesis, Department of Computer Science, University of Toronto.

Yu, E. (1997, jun). Towards modelling and reasoning support for early-phase requirements engineering. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97)*, Washington D.C., USA, pp. 226–235.

Yu, E. and L. Cysneiros (2002). Designing for privacy and other competing requirements. In *Proceedings of the 2nd Symposium on Requirements Engineering for Information Security (SREIS-02)*. Springer Verlag.

Yu, E., L. Lin, and J. Mylopoulos (2007). *A Social Ontology for Integrating Security and Software Engineering*, pp. 70–106. IGI Publishing.