Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT
Information Systems 29 (2004) 187–203
www.elsevier.com/locate/infosys

Designing information systems in social context: a goal and scenario modelling approach

Lin Liu[a,*], Eric Yu[b]

[a] Department of Computer Science, University of Toronto, 40 St. George St., Toronto, Ont., Canada M5S 2E4
[b] Faculty of Information Studies, University of Toronto, 140 St. George St., Toronto, Ont., Canada M5S 3G6

Received 31 August 2002; received in revised form 14 January 2003; accepted 23 February 2003

Abstract

In order to design a better information system, a designer would like to have notations to visualize how design experts' know-how can be applied according to one's specific social and technology situation. We propose the combined use of a goal-oriented requirements language (GRL) and a scenario-oriented notation Use Case Maps (UCM) for representing design knowledge of information systems. Goal-oriented modelling is used throughout the requirements and design process. In GRL, goals are used to depict business objectives and system requirements, both functional and non-functional. Tasks are used to represent different ways for achieving goals. Means-ends reasoning is used to explore alternative solutions and their operationalizations into implementable system constructs. Social context is modelled in terms of dependency relationships among agents and roles. Scenarios expressed in UCM are used to describe elaborated business processes or workflow. The complementary use of goal-oriented modelling with GRL and scenario modelling with UCM is illustrated with an example of designing a web-based training system.
© 2003 Elsevier Ltd. All rights reserved.

Keywords: Information system design; Goal-oriented requirements analysis; Scenario-based notation

1. Introduction

An information system is a social artifact serving the different interests of many stakeholders. Thus, inevitably, the design of an information system is a social activity, which involves understanding the social, organizational context of the system-to-be and making design decisions according to the limitations of environment and technology. As more and more software and information systems adopt Internet technologies and protocols for greater openness and interoperability, many new requirements appear. Unlike the closed computing environments for which most of the traditional information systems development methods were designed, the open, dynamic and almost unbounded nature of the Internet presents many new challenges and complexities. The design of new information systems, particularly Internet applications and web-based systems, are increasingly based on reusable components and flexible combination of existing patterns, which are hard to deal with without effective models and decision support tools.

*Corresponding author. Tel.: +1-416-978-7569; fax: +1-416-946-7132.
E-mail addresses: liu@cs.toronto.edu (L. Liu), yu@fis.utoronto.ca (E. Yu).

0306-4379/03/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S0306-4379(03)00052-8

In requirements engineering, a goal-oriented modelling approach has been recognized to be useful [1,2]. In general, goals describe the objectives that the system should achieve through the cooperation of actors in the software-to-be and in the environment [2]. It captures "why" the data and functions are there, and whether they are sufficient for achieving the high-level objectives that arise naturally in the requirements engineering process. The incorporation of explicit goal representations in requirement models provides a criterion for requirements completeness, i.e., the requirements can be judged as complete if they are sufficient to establish the goals that they are refining.

Scenario-oriented models present possible ways in which a system can be used to accomplish some desired functions or implicit purpose. Typically, it is a temporal sequence of interaction events between the intended software and its environment (composed of other systems and humans). A scenario could be expressed in various forms including narrative text, structured text, images, animation or simulations, charts, maps, etc. The content of a scenario could describe system–environment interactions or events inside a system. Scenarios have been used for various purposes—as means to elicit or validate system requirements, as concretization of use-oriented system descriptions, or as bases for test cases [3–5]. Scenarios have also become popular in other fields, notably human–computer interaction and strategic planning [6,7].

While goal modelling and scenario modelling each offers important capabilities, neither is adequate on its own for fully support requirements and design processes. Goals are sometimes abstract and implicit and can be complemented by concrete and explicit scenarios. Scenarios are usually partial and incomplete. Their inadequacies in coverage can be revealed through goal modelling and means-ends reasoning. Scenarios provide the snapshots of possible design solutions or fragments of solutions. Their concreteness facilitates the communication process between stakeholders and implementers of the system. On the other hand, goal modelling supports the explicit identification of alternatives and design tradeoffs. The proposed combined approach therefore draws on the complementary strengths of goals and scenarios to facilitate decision-making at all stages from early requirements to fairly detailed design. At the same time, it makes all the decision-making process traceable.

The goal-oriented requirements language (GRL) [8,9] is designed to support goal- and agent-oriented modelling and reasoning, providing guidance to the design process. In this paper, we propose the combined use of GRL with the scenario-based notation Use Case Maps (UCM) [10]. UCM allows the behavioral aspects of the designed system to be visualized at varying degrees of abstraction and levels of detail. The two notations complement each other to enable technical solutions to be described and elaborated, and evaluated according to their contributions to the objectives of different stakeholders, guiding the design towards viable solutions. While there are other ways of expressing scenarios, such as Use Cases and Activity Diagrams in UML, Message Sequence Charts, etc., we choose UCM to complement GRL due to the following considerations. UCM intends to straddle requirements and high-level architectural design stages, which closely matches with the scope of GRL. UCM supports the different levels of abstraction (with stub and plug-in mechanism) of system architecture, which complements the multi-level goal modelling of GRL.

Information systems design is a knowledge-intensive process. It involves domain-specific design knowledge, generic software design knowledge and knowledge about the specific situations of the current design. GRL and UCM together provide an ontology for expressing such knowledge. For example, consider the design of a web-based training (WBT) system. Domain-specific know-how on picking a lesson structure can be represented as UCM scenarios of common lesson structures. Generic software design knowledge on the possible collaboration mechanisms for a web-based system is captured as a GRL means-ends structure that connects the possible mechanisms (e-mail, newsgroup, chat, screen-sharing and audio/video conferencing) to the goal "Determine Collaboration Mechanism".

Basic concepts of GRL are introduced in Section 2. In Section 3, we summarize our approach of using GRL to incrementally model requirements and design. In Section 4, a case study in the e-training domain is used to illustrate the proposed approach. In Section 5, the combined use of GRL and UCM is introduced. In Section 6, related work is discussed. Conclusions and future work are in Section 7.

2. GRL modelling notation

The GRL [8,9] is a language for supporting goal- and agent-oriented modelling and reasoning about requirements, with an emphasis on dealing with non-functional requirements (NFRs) [11]. It provides constructs for expressing various types of concepts that are useful for supporting the requirements and high-level design process. There are three main categories of concepts: intentional elements, intentional links, and actors. GRL elements and links are intentional in that they are used in models that answer questions about intents, motivations and rationales, such as:

- Why are particular behaviors, information and structures are chosen to be included in the system requirements?
- What are the alternatives to be considered?
- What criteria are to be used to deliberate among alternative options?
- What are the reasons for choosing one alternative over others?

A GRL model can be composed of either a global goal model, or a series of goal models distributed amongst several actors. If a goal model includes more than one actor, then the intentional dependency relationships between actors can also be represented and reasoned about.

The intentional elements in GRL are goal, task, softgoal, resource and belief. A *goal* is a condition or state of affairs in the world that the stakeholders would like to achieve. A goal can be achieved in different ways, prompting alternatives to be considered. A goal can be either a business goal or a system goal. Business goals are about the business or state of the affairs the individual or organization wishes to achieve in the world. System goals are about what the target system should achieve, which, generally, describe the functional requirements of the target information system. In GRL graphical representation, goals are represented as a rounded rectangle with the goal name inside.

A *softgoal* is typically a quality (or non-functional) attribute on one of the other intentional elements. A softgoal is similar to a (hard) goal except that the criteria for whether a softgoal is achieved are not clear-cut and a priori. It is up to the developer to judge whether a particular state of affairs in fact sufficiently achieves the stated softgoal. NFRs, such as performance, security, accuracy, reusability, interoperability, time to market and cost are often crucial for the success of an information system. In GRL, NFRs are represented as softgoals and addressed as early as possible in the software lifecycle. They should be properly modelled and addressed in design reasoning before a commitment is made to a specific design choice. In the GRL graphical representation, a softgoal, which is "soft" in nature, is shown as an irregular curvilinear shape with the softgoal name inside.

A *task* specifies a particular way of doing something. It may be decomposed into a combination of sub-goals, sub-tasks, resources and softgoals. These sub-components specify a particular course of action while still allowing some freedom. Tasks are used to incrementally specify and refine solutions in the target system. They are used to achieve goals or to "operationalize" softgoals. These solutions provide operations, processes, data representations, structuring, constraints and agents in the target system to meet the needs stated in the goals and softgoals. In GRL graphical representation, tasks are represented as a hexagon with the task name inside.

A *resource* is a (physical or informational) entity, which may serve some purpose. From the viewpoint of intentional analysis, the main concern with a resource is whether it is available. Resources are shown as rectangles in GRL graphical representation.

The *Belief* construct is used to represent design assumptions and relevant environmental conditions. It allows domain characteristics to be considered and properly reflected in the decision-making process, hence facilitating later review, justification and change of the system, as well as enhancing traceability. Beliefs are shown as ellipses in GRL graphical representation.

Intentional links in GRL include means-ends, decomposition, contribution, correlation and dependency links. *Means-ends* links (—▷—) are used to describe how goals can be achieved. Each task connected to a goal by a means-ends link is one possible way of achieving the goal. *Decomposition* links (——+) define the sub-components of a task. A *contribution* link (→) describes the impact that one element has on another. A contribution can be negative or positive and can be of different extents. The extent is judged to be partial or sufficient based on Simon's concept of satisficing [12]. Accordingly, contribution link types include: *help* (positive and partial), *make* (positive and sufficient), *hurt* (negative and partial), *break* (negative and sufficient), *some+* (positive of unknown extent), *some−* (negative of unknown extent). *Correlation* links (dashed contribution links) describe the side effects of the existence of one element to others. *Dependency* links (—D—) describe the inter-agent dependent relationships.

An *actor* is an active entity that carries out actions to achieve its goals by exercising know-how. It is an encapsulation of intentionally, rationality and autonomy [13]. Graphically, an actor is represented as a circle, and may optionally have a dotted boundary, with intentional elements inside. To model complex relationships among social actors, we further define the concepts of agents (circle with a line at top), roles (circle with a line at bottom), and positions (four-leaf flower), each of which is an actor in a more specialized sense.

An *agent* is an actor with concrete, physical manifestations, such as a human individual or a machine. A *role* is an abstract characterization of the behavior of a social actor within some specialized context or domain of endeavor. A *position* is intermediate in abstraction between a role and an agent. It is a set of roles typically played by one agent. Positions can *cover* roles. Agents can *occupy* positions. Agents can also *play* roles directly. The "*INS*" construct represents the instance-and-class relation. The "*ISA*" construct expresses conceptual generalization/specialization.

3. A goal and scenario modelling design method

The proposed goal and scenario modelling approach was motivated by the need in the telecommunications domain for a notation for expressing and analyzing user requirements [14]. User requirements need to address behavior as well as quality attributes. While UCM is a useful requirement-level notation for telecommunications software [15], it does not provide systematic support for dealing with business objectives, goals, and NFRs during requirement analysis and their achievement during subsequent design. NFRs are requirements such as performance constraints, systems operational costs, reliability, maintainability, portability, interoperability, robustness, and the like. In software development practice, many NFRs are stated only informally, making them difficult to analyze, specify and enforce during software development and to be validated by the user once the final system has been built. Goals and NFRs, however, do play a crucial role during system development, serving as selection criteria for choosing among alternatives during requirements analysis, for example, determining where the system boundaries should be and what functional requirements to include in the system.

Many of the alternative approaches to deal with NFRs originated from the technical work related to quality metrics. Such approaches attempt to quantify NFRs and then measure to what extent an existing system or parts of it meet the desired NFRs. Useful metrics exist only for a small number of NFRs such as performance, reliability, software complexity, and development process maturity. Moreover, metric-based approaches are hard to use. During analysis and design there are many competing requirements, many of which are not quantitative. The GRL notation deals with NFRs and goals during the process of requirements analysis and system design; it allows for the expression of conflict between goals, of decisions that resolve conflicts and of the rationale for the trade-off decisions. The agent aspect of GRL helps in considering multiple stakeholders' concerns simultaneously.

To support early requirements engineering and high-level system design, our goal modelling approach aims to elicit, refine and operationalize customer-specific requirements incrementally based on the knowledge of domain experts, until a satisfactory design is found. In this process, the overall objectives of a system have to be clarified, the concrete behaviors and constraints of the system-to-be need to be elaborated, and functions should be assigned to responsible units in that system.

The goal- and agent-oriented modelling in GRL focuses on answering the "why" questions of requirements (such as "why does the system need to be redesigned?" or "why is the interface designed as it is?"). The strength of GRL modelling is that it puts the design in a broader context, it considers from different stakeholders' viewpoints, and seeking for a balanced solution for all. Another advantage of GRL is that not only functional requirements but also NFRs (in other words, the quality requirements) are dealt with.

While goal orientation can be highly useful for requirements engineering, goals are sometimes too abstract to capture all at once. Often they are discovered and become explicit only after a deeper understanding of the system has been achieved. In particular, users and developers often find it natural to think about operational scenarios about using the hypothetical system. More conventional requirements and design notations typically answers the "what" questions such as "what should the system do to provide activity centered electronic lessons?" or "what is the process of giving learner customized tutorial?"

The general steps of the proposed approach are illustrated in Fig. 1. From the flowchart, we can see that goal modelling and scenario modelling proceed in parallel, and they can interact at certain points in each round. In the goal-oriented modelling process, actor dependency models are first created, then the original business objectives and system requirements are identified and operationalized, until some concrete design options are obtained. These design options are explored with UCM scenarios. On the UCM side, business process or workflow, as well as responsibility assignment are visualized and analyzed. On both sides, new requirements may become evident by asking why questions, and be entered into the GRL model. When all scenarios are acceptable, and all goals and softgoals are sufficiently fulfilled, the solution fragments for each independent goal can be assembled to form a complete design for the intended system. Elaborated descriptions of use cases, processes and information flow are also obtained.

4. Case study: designing a web-based training system

The proposed goal and scenario modelling approach is best used to address cases where there are multi-stakeholders with diverse concerns and expectations, leading to complex interactions among functional and NFRs that need to be balanced and traded off. There should be well-established domain knowledge bases that the current design can benefit from. The complexity of the case should be such that there are many decision points at which multiple alternatives need to be considered, and where at least some of the alternatives can be visualized as scenarios. The proposed approach supports both the design of new systems, and the reengineering of legacy systems, as suggested by the iterative design process shown in the flowchart of Section 3.

To illustrate the application of the goal and scenario modelling approach, we use the example of designing a WBT system, adapted from [16]. Web-based information system usually involves multiple stakeholders with different interests. These stakeholders, modelled as intentional agents, impose complicated functional and quality requirements on the future system, which need to be considered and evaluated systematically according to the prospective solutions.

Starting from the identification of the major stakeholders of the domain, we explain in sequence how to capture the original business objectives of

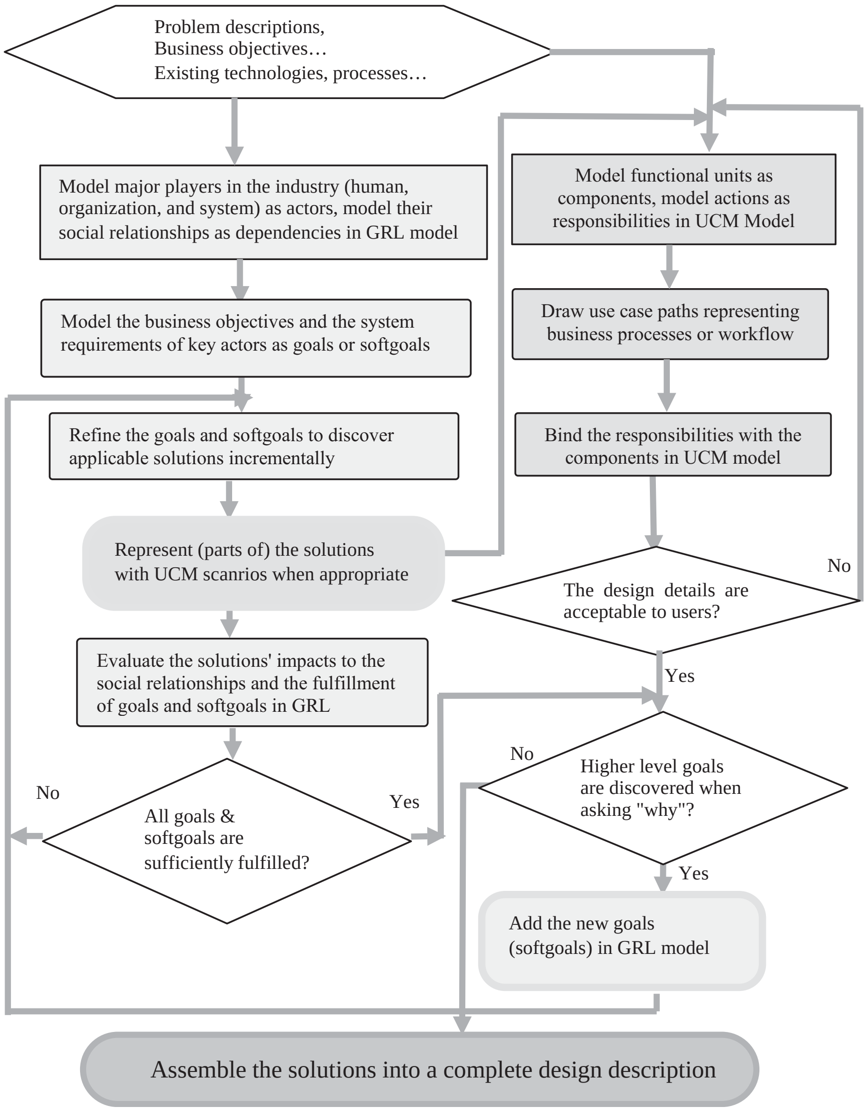
Problem descriptions,
Business objectives…
Existing technologies, processes…
Model major players in the industry (human, organization, and system) as actors, model their social relationships as dependencies in GRL model
Model the business objectives and the system requirements of key actors as goals or softgoals
Refine the goals and softgoals to discover applicable solutions incrementally
Represent (parts of) the solutions with UCM scanrios when appropriate
Evaluate the solutions' impacts to the social relationships and the fulfillment of goals and softgoals in GRL
All goals & softgoals are sufficiently fulfilled?
No
Yes
Model functional units as components, model actions as responsibilities in UCM Model
Draw use case paths representing business processes or workflow
Bind the responsibilities with the components in UCM model
The design details are acceptable to users?
No
Yes
Higher level goals are discovered when asking "why"?
No
Yes
Add the new goals (softgoals) in GRL model
Assemble the solutions into a complete design description

Fig. 1. Goal modelling based system design process.

the stakeholders, refine and operationalize these objectives into applicable design alternatives with GRL.

4.1. Step 1: modelling social entities and their relationships

Placing system design within its broader social context [17] (as in Fig. 3), the proposed modelling approach helps to address the following questions systematically: Who are the major players in the business domain? What are the generic relationships between these players? How to specialize these generic patterns through role-assignment and agent class instantiation? The major players are modelled as actors. The relationships between players are modelled as actor dependency relationships of different type. Then by distinguishing abstract roles and concrete agent classes and instances, we may model requirements at both

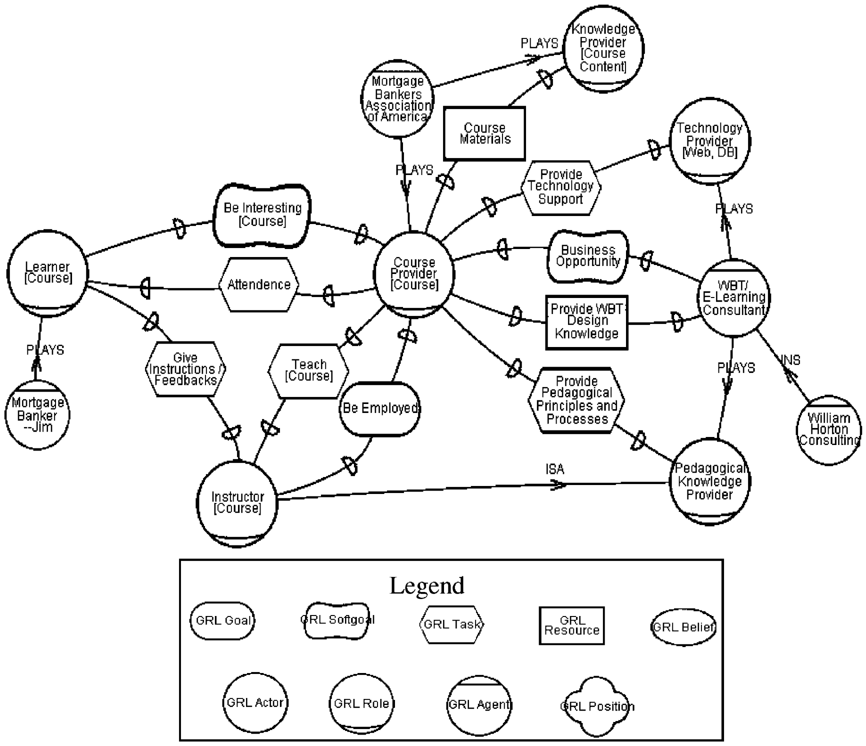
PLAYS
Knowledge Provider [Course Content]
Mortgage Bankers Association of America
Course Materials
Technology Provider [Web, DB]
PLAYS
Provide Technology Support
PLAYS
Be Interesting [Course]
Learner [Course]
Course Provider [Course]
Business Opportunity
Attendence
WBT/ E-Learning Consultant
Provide WBT Design Knowledge
PLAYS
Give Instructions / Feedbacks
Teach [Course]
Provide Pedagogical Principles and Processes
PLAYS
INS
Mortgage Banker --Jim
Be Employed
William Horton Consulting
ISA
Pedagogical Knowledge Provider
Instructor [Course]
Legend
GRL Goal
GRL Softgoal
GRL Task
GRL Resource
GRL Belief
GRL Actor
GRL Role
GRL Agent
GRL Position

Fig. 2. Major players in E-Learning domain, agent dependency relationships, role-playing relationships and agent classification.

the domain level (generic patterns) and the application level (specialization of the generic patterns).

In the WBT example, some of the major players are course provider, learner, technology provider, knowledge provider, and instructor. These are modelled as roles because they embody abstract capabilities and wants (Fig. 2). Learner depends on Instructor to Give Instructions and Feedback. Instructor depends on Course Provider to Be Employed. Course Provider depends on support from Technology Provider, Knowledge Provider and Pedagogical Knowledge Provider. Square brackets are used to include parameters necessary for identifying the actors. The agent WBT/E-Learning Consultant (a person) plays both the roles of Technology Provider and Pedagogical Knowledge Provider.

Apart from the three instance level agents—Mortgage Bankers Association of America, Mortgage Banker Jim, and William Horton Consulting, the model represents the common practices of the e-training domain, and is a reusable domain knowledge model. Mortgage Bankers Association of America plays the role of Course Provider as well as the role of Knowledge Provider, so it inherits all the dependency relationships of the two abstract roles in reality.

4.2. Step 2: modelling business objectives

After the main players are identified, their high-level business objectives will be elicited, i.e., what they hope to accomplish for their organization, their sponsors, or their financial backers by using the information system under consideration. Thus, these objectives and requirements will be modelled as primitive goals or softgoals of the actors. We use (hard) goals to represent functional requirements, and softgoals for NFRs.

In our case study, the Mortgage Bankers Association of America playing Course Provider has two specific targets in mind:

- Earn $200,000 by selling courses.
- Reduce costs of training by 50% over the next year.

In the initial GRL goal model in Fig. 3, they are represented as softgoals (we consider them as variations of the NFR "profitability"). From commonsense knowledge, we also know that a course provider's primitive goal is to provide course; thus, it is added as a goal of the corresponding role.

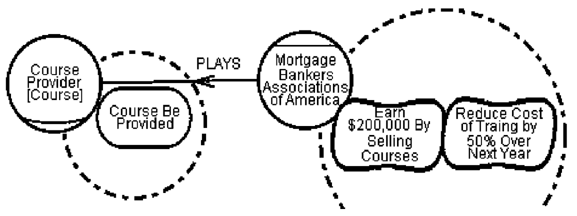
Course Provider [Course]
PLAYS
Mortgage Bankers Associations of America
Course Be Provided
Earn $200,000 By Selling Courses
Reduce Cost of Traing by 50% Over Next Year

Fig. 3. Business objectives represented as softgoals in original goal model.

4.3. Step 3: generating design alternatives

Starting from the initial goals and softgoals, we proceed to explore the alternative business processes, methods or technologies used in this industry to achieve these goals. A specific way for achieving a goal is represented as a task, and it is connected to the corresponding goal by a means-ends link, while being connected to a softgoal by contribution links. When refining a high-level goal/softgoal, we may use decomposition, specialization, substitution, or other refinement techniques applicable to the domain, until operational design solutions are found [11].

In Fig. 4, the two softgoals of the Mortgage Banker's Association of America can be reduced into two general softgoal applicable to all Course Providers—Low Cost and Customer Satisfaction. The two softgoals, together with the goal of Course Be Provided, are refined individually. Since the two most obvious choices for giving a course are to provide Conventional Classroom Training, or WBT, a first design choice is made between them. From the two initial softgoals, we can see that cost is more critical for the stakeholder. Thus, the softgoal Low Cost is refined in detail.

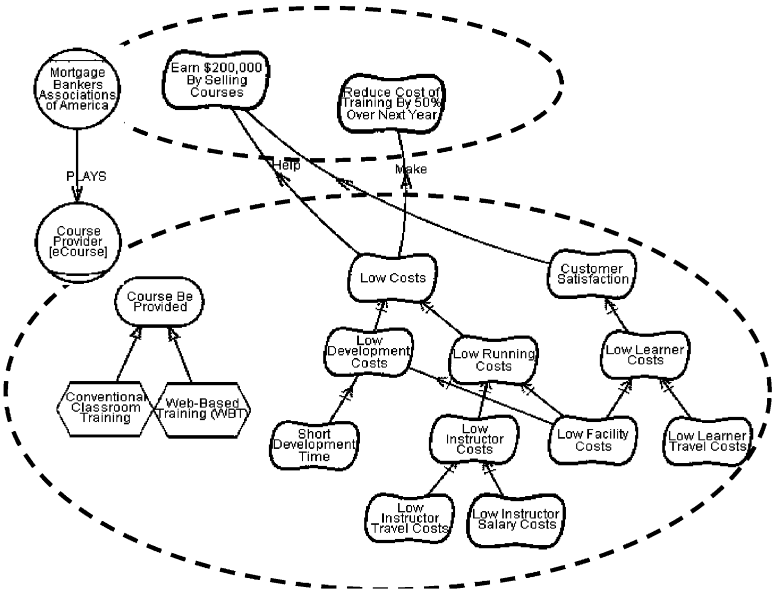
Mortgage Bankers Associations of America
Earn $200,000 By Selling Courses
Reduce Cost of Training By 50% Over Next Year
Help
Make
PLAYS
Course Provider [eCourse]
Low Costs
Customer Satisfaction
Course Be Provided
Low Development Costs
Low Running Costs
Low Learner Costs
Conventional Classroom Training
Web-Based Training (WBT)
Short Development Time
Low Instructor Costs
Low Facility Costs
Low Learner Travel Costs
Low Instructor Travel Costs
Low Instructor Salary Costs

Fig. 4. Explore possible designs for the future system (high level).

Table 1
Cost estimation on the two kinds of training

	Develop time (h)	Develop cost ($/h)	Instructor travel cost ($)	Instructor salary cost ($/student)	Facility cost ($/student)	Learner travel cost ($)	Total estimate cost ($)
Conventional classroom training	50	50	1500	25	500	1500	513,000
Web-based training	200	100	0	50	50	0	338,500

4.4. Step 4: evaluating design alternatives: contributions to softgoals

To evaluate how the design alternatives are serving the specific business objectives and the quality expectations of stakeholders, contributions of the design options to the softgoal will be explicitly modelled. In addition to analyzing the solutions within the boundary of one actor, we can also evaluate the two solutions according to their impacts to the relationships among actors. This will be illustrated in Step 7.

In Table 1, we list how the two solutions WBT and Conventional Classroom Training (represented as task nodes) lead to different cost on the items indicated by the sub-softgoals of Low Cost in Fig. 4. Based on these data, in Fig. 5, we use contribution links to depict that WBT *helps* the goal of Lower Total Training Costs, which in turn *help*s the satisficing of Reduce Cost of Training by 50% Over Next Year. Conventional Classroom Training *hurts* the fulfillment of this goal. Furthermore, the fulfilling of this goal *helps* the achievement of Earn $200,000 By Selling Courses. The result of this initial analysis suggests that WBT may be a better option for the current stakeholder. The upper part of this model (the two softgoals and the *help* relationship between them) is only applicable to the current system, while the lower part (the structure showing the different resource consumption of the two solutions) depicts generic domain knowledge reusable for all course providers of WBT system.

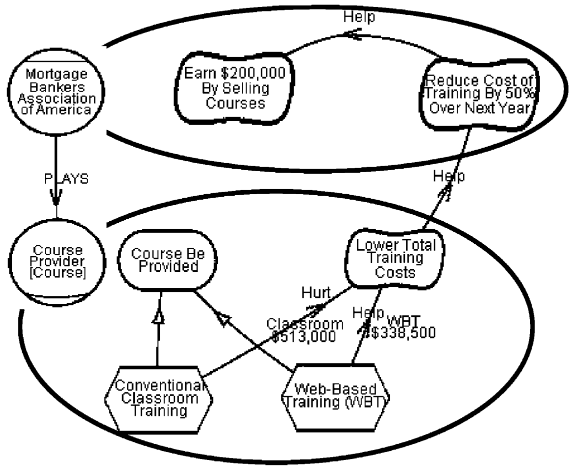
Help
Mortgage Bankers Association of America
Earn $200,000 By Selling Courses
Reduce Cost of Training By 50% Over Next Year
PLAYS
Help
Course Provider [Course]
Course Be Provided
Lower Total Training Costs
Hurt
Classroom $513,000
Help
WBT $338,500
Conventional Classroom Training
Web-Based Training (WBT)

Fig. 5. Compare alternative designs by resource consumption.

4.5. Step 5: elaborating on the candidate solution

Having selected one solution over the other, we need to evaluate the advantages and disadvantages of the candidate solution further. In this round of evaluation, other softgoals of concern are considered, to whom the candidate solution's contributions are investigated.

GRL evaluates the satisfaction of a softgoal via a qualitative labelling procedure [11]. The label of a high-level node is computed from the label of low-level nodes and the type of contribution from these nodes, with possible user input. As one can hardly find a perfect technology, or a perfect situation that a technology can apply to without any change, a best solution, for many needs, may be a hybrid combining the best features of different solutions. In this case, alternative solutions need to be further decomposed and reassembled. For disadvantages indicated by negative links or labels, mitigation measures are sought to strengthen the current solution. The labels are defined as follows: *Satisficed* (✓), *Weakly Satisficed* (✓•), *Conflict/irresolvable* (⚡), *Undecided* (?), *Weakly Denied* (✗•), *Denied* (✗).

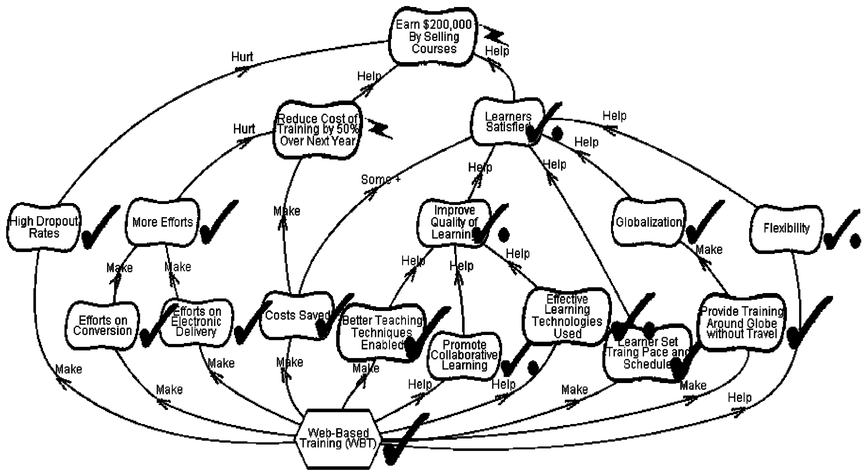
Earn $200,000 By Selling Courses
Reduce Cost of Training by 50% Over Next Year
Learners Satisfied
High Dropout Rates
More Efforts
Improve Quality of Learning
Globalization
Flexibility
Efforts on Conversion
Efforts on Electronic Delivery
Costs Saved
Better Teaching Techniques Enabled
Promote Collaborative Learning
Effective Learning Technologies Used
Learner Set Traing Pace and Schedule
Provide Training Around Globe without Travel
Web-Based Training (WBT)
Hurt
Help
Make
Some +

Fig. 6. Evaluate candidate design's advantages and disadvantages.

The corresponding goal model in Fig. 6 shows that the advantages of WBT include Costs Saved, Better Teaching Techniques Enabled, Collaborative Learning Promoted, and Effective Learning Technologies Used. Consequently, Quality of Learning Improved is *weakly satisficed* (represented with a check mark with a dot underneath). It also contributes positively to Globalization, Flexibility, and both *help* the Learner's Satisfied, as the right-hand side of the model suggests. On the left side, unfortunately, negative contributions are also revealed, e.g., the inherent High Dropout Rates and More Efforts on Conversion and Electronic Delivery of WBT *hurts* the high-level softgoals of the stakeholder. These negative contributions make the two high-level softgoals be labelled as *irresolvable* (denoted by a thunderbolt symbol)—there are both strong positive contributions and strong negative contributions.

To weaken the negative contributions, countermeasures such as Require Commitment are added in the design. They are represented as tasks with negative correlation links (the dotted lines with arrows) to the unfavorable contributions in the graph. Adding these countermeasures will weaken the impact of the softgoals with negative contributions. In such a case, although the contributions from these softgoals are still *undecided*, high-level softgoals are already judged as *weakly satisficed* based on the system developer's opinion (Fig. 7).

4.6. Step 6: refining a solution

After each decision-making session, the design proceeds further by identifying the essential sub-processes/components of the candidate solution, then steps 3, 4 and 5 will be repeated. Sub-components are connected to the root task with decomposition links.

The model in Fig. 8 illustrates how the task Build a WBT system is refined. First of all, a Course Provider needs to Choose e-Course Pattern, decide whether to use Collaboration Mechanisms and what mechanism to use, and Pick a Lesson Structure for the course. As all of these sub-processes are necessary steps for the finishing of the root task, they are represented as sub-goals connected to the root task with decomposition links. Similarly, existing collaboration mechanisms are connected to the goal Determine Collaboration Mechanisms with means-ends links. Their impacts to social dependencies and contributions to course provider's business objectives will be further explored. By making tradeoffs among the possible solutions, one iterates until an acceptable design is obtained.

4.7. Step 7: evaluating impacts on dependencies

Now we come to the decision-making process for Choose e-Course Pattern. In Fig. 8, two

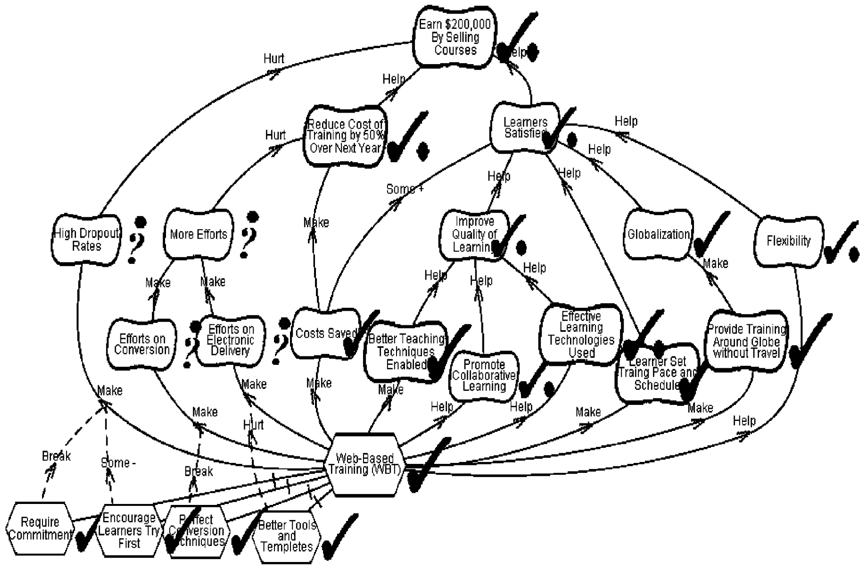
Earn $200,000 By Selling Courses
Reduce Cost of Training by 50% Over Next Year
Learners Satisfied
High Dropout Rates
More Efforts
Improve Quality of Learning
Globalization
Flexibility
Efforts on Conversion
Efforts on Electronic Delivery
Costs Saved
Better Teaching Techniques Enabled
Promote Collaborative Learning
Effective Learning Technologies Used
Learner Set Traing Pace and Schedule
Provide Training Around Globe without Travel
Web-Based Training (WBT)
Require Commitment
Encourage Learners Try First
Perfect Conversion Techniques
Better Tools and Templetes
Hurt
Help
Make
Some +
Some -
Break

Fig. 7. Install mitigation measures to the design.

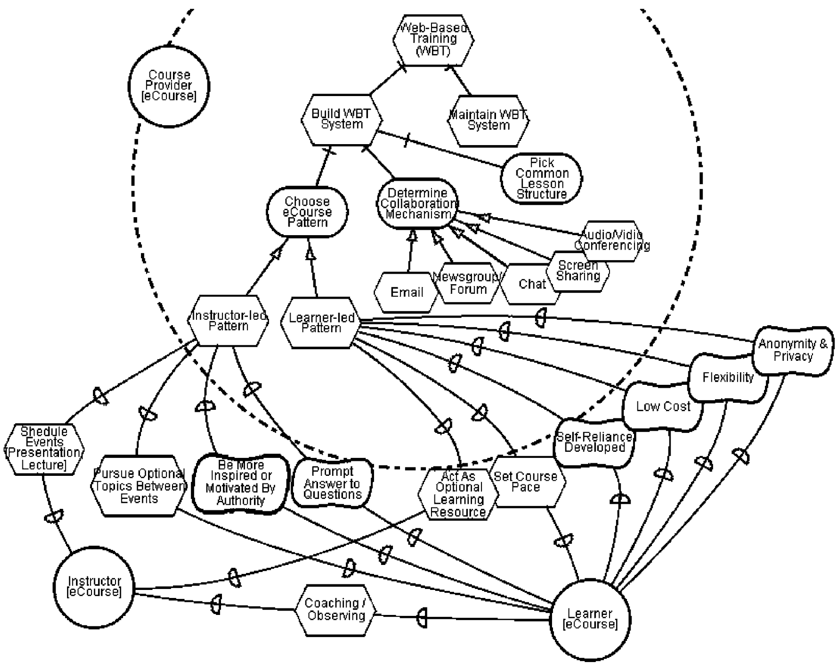

Fig. 8. Refinement of design and decision-making based on social relations.

alternatives are generated and connected to the parent node with means-ends link: Instructor-led Pattern and Learner-led Pattern. In addition to the kind of analysis shown in step 4, we can also evaluate the alternative solutions according to their impacts to social relationships. The dependency links pointing from these two tasks tell us that the two solutions lead to quite different role characteristics—the Instructor is the driving force in Instructor-led Pattern, but only Act As an Optional Learning Resource in Learner-led Pattern. Conversely, the dependencies pointing to the two task nodes show that they have different capabilities and qualities to offer. Learner-led Pattern favors Lower Cost. Learner enjoys more Flexibility on their schedule and learning content, and they also appreciate the Anonymity and Privacy. In Instructor-led Pattern, Instructor can often Prompt Answer to Questions, and the Learner Be More Inspired or Motivated. Thus, corresponding to the requirements of different kinds of courses, different pattern can be adopted.

5. Scenario-based analysis

As the goal-oriented design proceeds, finer-grained analysis needs to be conducted. The scenario-based notation UCM comes into use.

5.1. Use case maps

UCM [10] provide a visual notation for scenarios, which is proposed for describing and reasoning about large-grained behavior patterns in systems, as well as the coupling of these patterns. The UCM notation employs scenario paths to illustrate causal relationships among responsibilities. It provides an integrated view of behavior and structure by allowing the superimposition of scenario paths on a structure of abstract components. Scenarios in UCM can be structured and integrated incrementally. This enables reasoning about and detection of potentially undesirable interactions between scenarios and components.

Basic elements of UCMs are start points, responsibilities, end points and components. *Start points* (filled circles) represent pre-conditions or triggering causes. *End points* (bars) represent post-conditions or resulting effects. *Responsibilities* (crosses) represent actions, tasks or functions to be performed. *Components* (boxes) represent entities or objects composing the system. *Use case Paths* (wiggle lines) connect start points, responsibilities and end points. A responsibility is bound to a component when the cross is inside the component. In this case, the component is responsible for performing the action, task, or function represented by the responsibility.

When maps become too complex to be represented as a single UCM, a mechanism for defining and structuring sub-maps becomes necessary. A top level UCM, referred to as a root map, can include containers (called stubs) for sub-maps (called plug-ins). Stubs are represented as diamonds. Stubs and plug-ins are used to solve the problems of layering and scaling or the dynamic selection and switching of implementation details. Other notational elements include OR-join, OR-fork, AND-join, AND-fork, timer, abort, failure point, and shared responsibilities. A detailed introduction to and examples of these concepts can be found in [10].

5.2. Combined use of GRL and UCM

Although UCM can represent system designs in a high-level way, the tradeoffs between alternatives, and the intentional reasoning behind design decisions cannot be explicitly shown. In our approach, we couple GRL with UCM to provide support for reasoning about scenarios by establishing correspondences between intentional GRL elements and functional components and responsibilities in the scenario models of UCM. The complementary modelling of goals and scenarios aid in identifying further goals and additional scenarios (and scenario fragments) important to system design, thus contributing to the completeness and accuracy of requirements, as well as to the quality of system design.

Continuing with the design of WBT system in Section 4, we now consider the implementation of the goal Pick Lesson Structure. The alternative structures are denoted as task nodes in the bottom of the GRL model in Fig. 9. It is hard to tell which structure is more appropriate only by doing strategic, intentional analysis with GRL. In order to visualize the behavioral aspects of the alternatives, we link the appropriate GRL nodes to scenarios in UCM.

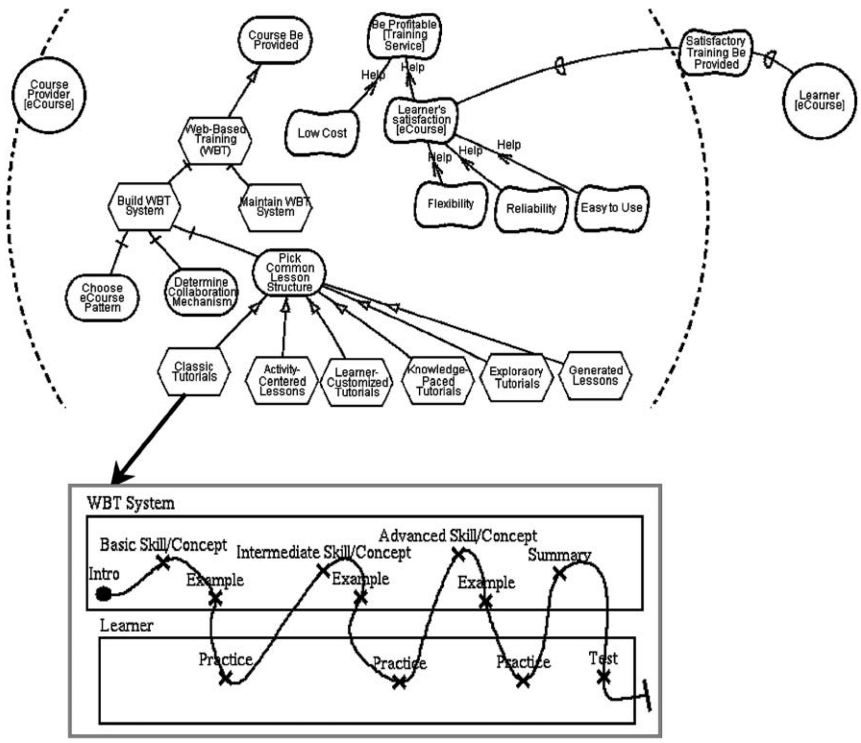

Fig. 9. Design alternatives and the corresponding scenarios.

In the lower half of Fig. 9 a class structure representing Classic Tutorials is depicted as a UCM scenario. In the scenario, WBT system and Learner are represented as agent components (rectangles), which holds responsibilities (small crosses along the wiggle lines). The scenario shows that, in a classic tutorial, after an introduction, learners do readings through a series of sessions, each teaching a more difficult concept or skill. At the end of the sequence (denoted with a use case path, the wiggle line with filled circle head, and small bar tail) is a summary and a test. Examples and practice are also provided in each session.

Elaborating on these details helps the identification of new requirements. For example, Learner's Satisfaction, Flexibility, Reliability and Easy to Use are required for the training program to be profitable and successful. Thus, these newly identified requirements are added on the right-hand side of the goal model in Fig. 9.

In Fig. 10, the class structure is evaluated using the qualitative evaluation procedure mentioned above. The result shows that the current structure is not an ideal choice. It is simple, reliable, but lack flexibility.

Thus, a good-to-have feature of the above class structure is that, for each learner, the tutorial is as easy and straightforward as a class tutorial, but the course content can be customized by different learners. Below is a scenario for this class structure—Learner Customized Tutorial.

The scenario in Fig. 11 shows that the use case path branches for different learners if they choose different subjects in the course, from which we can see that student's Individual/Specific Needs Be Considered. This class structure satisfied all the currently required softgoals, so it is a possible choice for the designer.

In this case study, the UCM models are rather simplistic because we have only tackled the highest

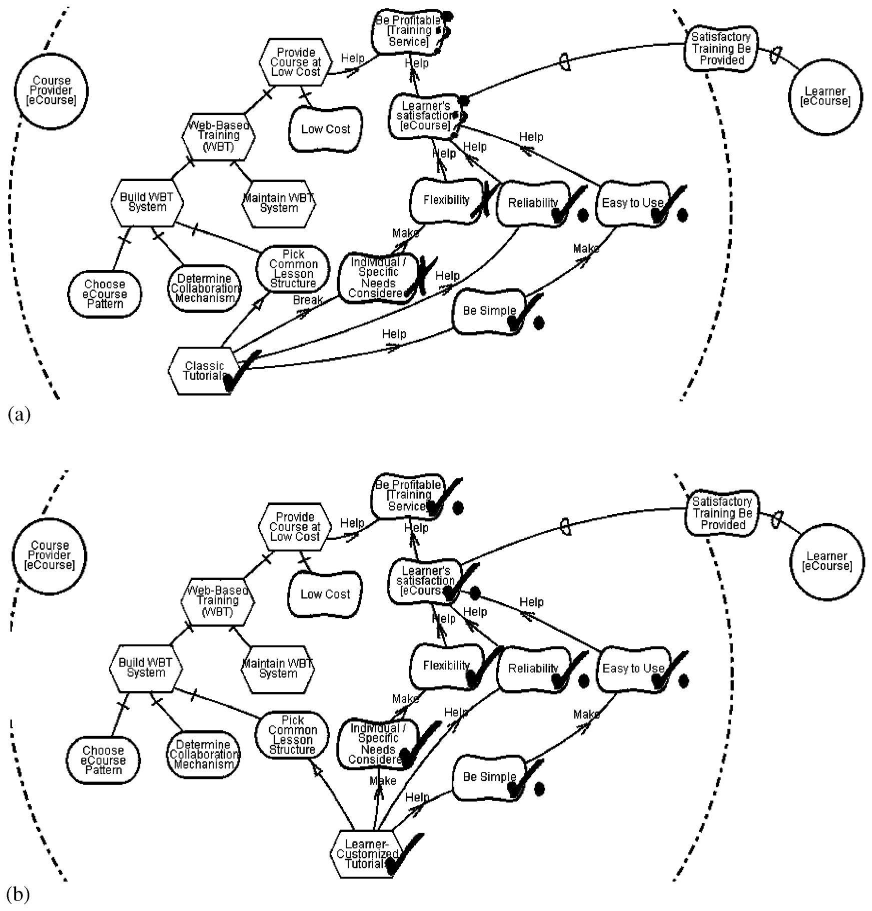
Course Provider [eCourse]
Be Profitable [Training Service]
Provide Course at Low Cost
Help
Satisfactory Training Be Provided
Learner [eCourse]
Low Cost
Learner's satisfaction [eCourse]
Web-Based Training (WBT)
Build WBT System
Maintain WBT System
Flexibility
Reliability
Easy to Use
Make
Choose eCourse Pattern
Determine Collaboration Mechanism
Pick Common Lesson Structure
Individual / Specific Needs Considered
Break
Be Simple
Classic Tutorial
Learner-Customized Tutorial
(a)
(b)

Fig. 10. Evaluation of tutorial structure: (a) classical and (b) learner customized.

level of process design, and the processes in e-training are not very complicated. As we go down to a sufficiently detailed design, a UCM model can be fairly complex, and more modelling constructs need to be used. Having analyzed the benefits and tradeoffs of these structures, we can see that UCM is a useful complement to GRL in the process from requirements to high-level design. It provides a concrete model of each design alternative.

During each step shown above, new NFRs may be detected and added to the GRL model. At the same time, in the GRL model, new means to achieve the functional requirements can be explored and concretized in a UCM model. Thus, the above design process may iterate until an acceptable design is reached.

6. Discussion and related work

The above example illustrated some of the benefits of coupling goals and scenarios during requirements analysis and design. GRL and UCM together facilitates the transition from a requirements specification to a high-level design involving the consideration of alternative architectures and the discovery of further requirements that must be vetted by the stakeholders. Both of the notations have dynamic refinement capabilities. During refinement, a high level of abstraction is maintained, as scenarios in UCM are described as first class entities without requiring reference to system sub-components, specific inter-component communication facilities, or sub-component states

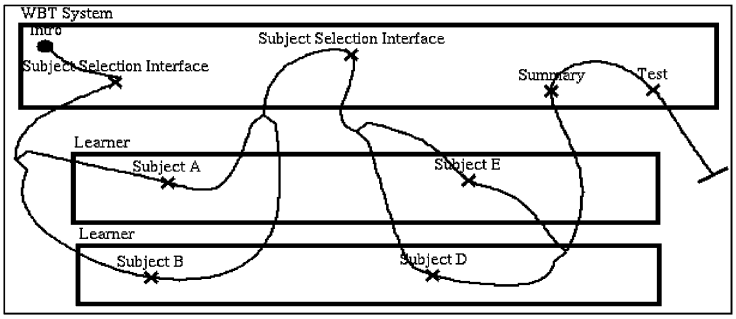

Fig. 11. UCM scenario for Learner Customized Tutorial.

[10]. As one allocates scenario responsibilities to architectural components in UCM, GRL helps keeping track of decisions at various stages. GRL provides facilities to express, analyze and deal with goals and NFRs. It also provides facilities to capture reusable analysis and design knowledge related to know-how for addressing NFRs and to manage evolving requirements.

The work of this paper builds on an original submission to ITU-T Study Group 10 (now Study Group 17) on the topic of User Requirements Notation (URN) [14]. The URN is intended to allow software engineers to specify, review for correctness, and possibly discover requirements for a proposed new system or for extensions to an existing system. UCM is being proposed for specifying functional requirements, and GRL for NFRs. The methodology introduced in this paper illustrates how the two modelling notations complement each other.

Goal-oriented modelling has received considerable attention with requirements engineering [2]. The KAOS approach is most concerned with the generation of alternative system designs from high-level goals defined in temporal logic [1]. In [18], the process of inferring formal specifications of goals and requirements from scenario descriptions is studied. While goal elaboration and scenario elaboration are treated as intertwined processes, the focus of the work is mainly on goal elicitation. Our emphasis is the other way around, i.e., how to use goal model (especially NFRs) to direct design based on scenarios as well as other notations. The fundamental point is that the goal-oriented modelling and its interaction with other modelling activity run through requirements to the entire design process.

In the CREWS project, Rolland et al. [19] have proposed the coupling of goals and scenarios in requirements engineering with CREWS-L'Ecritoire. In CREWS-L'Ecritoire, scenarios are used as a means to elicit requirements/goals of the system-to-be. Both goals and scenarios are represented as structured text. The coupling of goal and scenario could be considered as a "tight" coupling, as goals and scenarios are structured into <Goal, Scenario> pairs, which are called "requirement chunks". The focus is mainly on the elicitation of functional requirements and goals. In GRL, both functional and NFRs are considered, with special attention being paid to NFRs. The modelling process involves both requirements engineering activities and high-level architectural and process design.

Also in the CREWS project, Haumer et al. [20] have proposed to use real world scenes to elicit and validate requirements specifications. Goals are used as central concepts of requirement description. Hierarchical goal structures are linked and annotated with positive and negative scenarios. Their approach typically starts from fairly low-level functional goals rather than high-level goals like "increase profit by 10%". The kinds of scenarios they propose to capture are multi-media scenarios of current system usage.

The Software Architecture Analysis Method (SAAM) [21] is a scenario-based method for evaluating architectures. It provides a means to characterize how well a particular architectural design responds to the demands placed on it by a particular set of scenarios. Based on the notion of context-based evaluation of quality attributes, scenarios are used as a descriptive means of specifying and evaluating quality attributes. SAAM scenarios are use-oriented scenarios, which are designed specifically to evaluate certain quality attributes of architecture. The evaluations are done using simulations or tests on a finished design. In the GRL and UCM approach, scenarios are more design oriented, being concerned with the refinement of system requirements. The quality of the architectures corresponding to these scenarios is judged based on expert knowledge as the design proceeds.

7. Conclusions and future work

The complementary of GRL and UCM supports the progress from abstract requirements, both functional and non-functional, to concrete system models. The approach combines an intentional strategic actor's view of design rationales and a non-intentional behavioral view of the future system. We believe the approach is useful to information systems in general, where there are conflicting goals and tradeoffs to be dealt with during design. A case study in telecommunication domain is discussed in [22], which focuses more on using goal and scenario together in software architectural design. Combining the two notations may not be necessary for some classes of applications. For example, if the design of a system does not decide on temporal orders, causal relationships, and other behavioral characteristics, then GRL is sufficient. On the other hand, if during the design of the system, there are not many alternatives and competing goals, and the main task for the software engineer is just to work out all the details, then UCM itself may be quite enough for the work.

For future work, it would be worthwhile to investigate tighter coupling at language level to provide more guidance and support. In the current approach, the coupling of goals with scenarios is loose—goal models and scenario models can be constructed fairly independently. One scenario model may refer to more than one goal, and vice versa. There are no rigid constraints on the requirements engineering and design process. That is, the goal model and behavior models can be developed in parallel simultaneously, interacting whenever there are design decisions to be traded off, or new design alternatives need to be sought, or new business goals or non-functional requirements are discovered.

GRL and UCM are vehicles for expressing knowledge. To make better use of GRL and UCM concepts, we need to acquire and accumulate both software design knowledge and more knowledge of various domains, and represent this knowledge in the corresponding modelling structures. The development of such repositories would enable the reuse of knowledge and provide useful guidance for the design process.

Another ongoing work is to extend a formal goal-oriented requirements language, Formal Tropos, so that the temporal properties shown in the UCM behavior models currently can be embedded into the goal models and so that they can be validated with model-checking techniques [23].

Acknowledgements

The work of this paper is motivated by an original submission to ITU-T study group 17 on the topic of User Requirements Notation (URN). The kind cooperation of members of Mitel Networks, Nortel Networks and other institutions is gratefully acknowledged. This work received financial support from NSERC, CITO, and Mitel Networks.

References

[1] A.V. Lamsweerde, Requirements engineering in the year 00: a research perspective, in: The Proceedings of the 22nd International Conference on Software Engineering, Limerick, June 2000, ACM press, New York.

[2] E. Yu, J. Mylopoulos, Why goal-oriented requirements engineering, in: E. Dubois, A.L. Opdahl, K. Pohl (Eds.), Proceedings of the Fourth International Workshop on Requirements Engineering: Foundations of Software Quality, Pisa, Italy, Presses Universitaires de Namur, Paris, June 1998, pp. 15–22.

[3] J.G. Leite, J. Doorn Hadad, G. Kaplan, A scenario construction process, Requirements Eng. 5 (1) (2000) 38–61.

[4] A. Sutcliffe, N. Maiden, S. Minocha, D. Manual, Supporting scenario-based requirements engineering, IEEE Trans. Software Eng. 24 (12) (1998) 1072–1088.

[5] K. Weidenhaupt, K. Pohl, M. Jarke, P. Haumer, Scenario management in software development: current practice, IEEE Software (15) (1998) 34–45.

[6] J.M. Carroll, Introduction: the scenario perspective on system development, in: J.M. Caroll (Ed.), Scenario-Based Design: Envisioning Work and Technology in System Development, Wiley, 1995, pp. 1–17.

[7] M. Jarke, X.T. Bui, J. MC arroll, Scenario management—an interdisciplinary approach, Requirements Eng. J. 3 (3–4) (1998) 155–173.

[8] GRL web site. http://www.cs.toronto.edu/km/GRL/

[9] E. Yu, Towards modelling and reasoning support for early phase requirements engineering, in: Proceedings of the Third IEEE International Symposium on Requirements Engineering (RE '97), Washington, DC, USA, January 6–8, 1997, pp. 226–235.

[10] R.J.A. Buhr, Use case maps as architectural entities for complex systems, in: Transactions on Software Engineering, Vol. 24, No. 12, IEEE Press, New York, December 1998, pp. 1131–1155.

[11] L. Chung, B.A. Nixon, E. Yu, J. Mylopoulos, Non-Functional Requirements in Software Engineering, Kluwer Academic Publishers, Dordrecht, 2000.

[12] A H. Simon, The Sciences of the Artificial, 2nd Edition, The MIT Press, Cambridge, MA, 1981.

[13] E. Yu, Agent orientation as a modelling paradigm, Wirtschaftsinformatik 43 (2) (2001) 123–132.

[14] Z.150: Users requirements notation, Recommendation to ITU-T Study Group 17, Available at URN web site. http://www.usecasemaps.org/urn/

[15] D. Amyot, G. Mussbacher, N. Mansurov, Understanding existing software with use case map scenarios, in: Third SDL and MSC Workshop (SAM '02), Aberystwyth, UK, June 2002.

[16] W. Horton, Designing Web-Based Training, Wiley, New York, 1990.

[17] E. Yu, Agent-oriented modelling: software versus the world, in: The Proceedings Agent-Oriented Software Engineering AOSE-2001 Workshop, LNCS 2222, On-line at: http://www.fis.utoronto.ca/faculty/yu

[18] A.V. Lamsweerde, L. Willemet, Inferring declarative requirements specifications from operational scenarios, in: IEEE Transactions on Software Engineering, Special Issue on Scenario Management, December 1998.

[19] C. Rolland, G. Grosz, R. Kla, Experience with goal-scenario coupling in requirements engineering, in: Proceedings of the IEEE International Symposium on Requirements Engineering, 1998, Limerick, Ireland, June 1999.

[20] P. Haumer, K. Pohl, K. Weidenhaupt, Requirements elicitation and validation with real world scenes, IEEE Trans. Software Eng. 24 (12) (1998) 1036–1054.

[21] R. Kazman, L. Bass, G. Abowd, M. Webb, SAAM: a method for analyzing the properties of software architectures, in: Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italy, 1994, pp. 81–90.

[22] L. Liu, E. Yu, From requirements to architectural design—using goals and scenarios, in: ICSE-2001 Workshop: From Software Requirements to Architectures (STRAW 2001), May 2001, Toronto, Canada, pp.22–30. Toronto, Canada, May 14, 2001, On-line at: http://www.cs.toronto.edu/~liu/

[23] A. Fuxman, M. Pistore, J. Mylopoulos, P. Traverso, Model checking early requirements specifications in Tropos, in: Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, Toronto, Canada, August 2001, pp. 174–181.