Designing Web-Based Systems in Social Context: A Goal and Scenario Based Approach

Lin Liu and Eric Yu

Faculty of Information Studies, University of Toronto
{liu,yu}@fis.utoronto.ca

Abstract. In order to design a better web-based 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 language GRL and a scenarios-oriented notation UCM for representing design knowledge of web-based systems and information systems in general. Goals are used to depict business objectives, functional and non-functional system requirements. Tasks are used in the exploration of alternative technologies and their operationalizations into system constructs. Actors are used to do role-based analysis on social relationships. Scenarios are used to describe elaborated business processes or workflow. The approach is illustrated with an example of designing a web-based training system.

1 Introduction

In the context of requirements engineering and system design, goal-driven and scenario-based approaches have proven useful [9]. In order to overcome some of the deficiencies and limitations of these approaches when used separately, proposals have been made to couple goal, scenario and agent concepts together to guide the system design process. As there are both overlaps and gaps between these approaches, their interactions can be complex and highly dynamic.

In general, goals describe the objectives that the system should achieve through the cooperation of agents in the software-to-be and in the environment [14]. It captures „why" the data and functions are there, and whether they are sufficient for achieving the high-level objectives that arises 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 they are refining.

Scenarios 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 or humans). A scenario could be expressed in forms such as narrative text, structured text, images, animation or simulations, charts, maps, etc. The content

A. Banks Pidduck et al. (Eds.): CAISE 2002, LNCS 2348, pp. 37-51, 2002.

of a scenario could describe either system-environment interactions or events inside a system. Purpose and usage of scenarios also varies greatly. It could be used as means to elicit or validate system requirements, as concretization of use-oriented system descriptions, or as basis for test cases. Scenarios have also become popular in other fields, notably human-computer interaction and strategic planning [2].

A successful design relies on the clarity of user requirements and the close matching of requirements and the adopted technologies. In this paper, we explore the combined use of a goal-oriented notation GRL [5] and a scenario-based notation UCM [1] in early requirements engineering and system design. The GRL language is used to support goal and agent-oriented modelling and reasoning, providing guidance to the design process. The scenario orientation of the UCM notation allows the behavioral aspects of the designed system to be visualized at varying degrees of abstraction and levels of detail. Combining the two notations makes it possible to evaluate technical solutions according to their contributions to the objectives of different stakeholders, guiding the design towards viable solutions.

Information system design is a knowledge-intensive process. It involves domain-specific 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, domain-specific know-how on picking a lesson structure is represented as UCM scenarios of common lesson structures. Generic software design knowledge on the possible collaboration mechanisms for a web-based system is depicted 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“.

In the next section, basic concepts of GRL and UCM are introduced. In section 3, we summarize our approach of using GRL and UCM together to incrementally modelling requirements and design. In section 4, a case study in the e-training domain is used to illustrate the proposed approach. In section 5, related work is discussed. Conclusions and future work are in section 6.

2 Modelling Notations

2.1 GRL

The Goal-oriented Requirements Language (GRL) [5] is a language for supporting goal and agent oriented modelling and reasoning about requirements, especially for dealing with non-functional requirements (NFRs)[3][14]. It provides constructs for expressing various types of concepts that appear during 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 particular behaviors, information and structures are chosen to be included in the system requirements?
- What alternatives are considered?

- What criteria are used to deliberate among alternative options?
- What are the reasons for choosing one alternative over others?

A GRL model can either be composed of 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. In general, how the goal is to be achieved is not specified, allowing 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 business affairs the individual or organization wishes to achieve. 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 the criteria for whether a softgoal is achieved is not clear-cut. It is up to the developer to judge whether a particular state of affairs in fact sufficiently achieves the stated softgoal. Non-functional requirements (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, non-functional requirements are represented as softgoals and addressed as early as possible in the software lifecycle, and be properly reflected in design decisions before a commitment is made to a specific implementation. 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 consist of subgoals, subtasks, 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, about which the main concern is whether it is available. Resources are shown as rectangles in GRL graphical representation.

Belief is used to represent design assumptions and environmental conditions. Beliefs make it possible for 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 are in fact achieved. Each task connected to a goal by a means-ends link is an alternative way to achieve the goal. *Decomposition* links define the sub-components

of a task. A *Contribution* link describe the impact that one element has on another. A contribution can be negative or positive. The extent of the contribution can be partial or sufficient based on Simon's concept of satisficing [12]. *Correlation* links describe the side effects of the existence of one element to others. *Dependency* links describe the inter-agent dependent relationships. Following are the graphical representations for links.

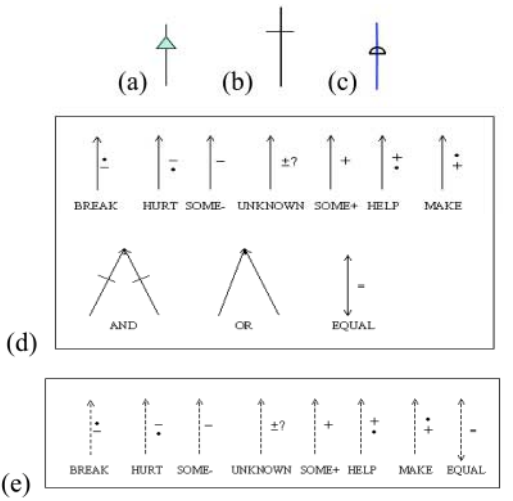

Fig. 1 (a) Means-Ends; (b) Decomposition; (c) Dependency; (d) Contribution; (e) Correlation

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 [16]. Graphically, an actor may optionally have a boundary, with intentional elements inside. To model complex relationships among social actors, we further define the concepts of agents, roles, and positions, 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. 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, and agents can also *play* roles directly. The „*INS*" construct represents the instance-and-class relation. The „*ISA*" construct expresses conceptual generalization/ specialization.

2.2 UCM

Use Case Maps (UCM)[1] 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* are filled circles representing pre-conditions or triggering causes. *End points* are bars representing post-conditions or resulting effects. *Responsibilities* are crosses representing actions, tasks or functions to be performed. *Components* are boxes representing entities or objects composing the system. *Use case Paths* are wiggle lines that connect start points, responsibilities and end points. A responsibility is said to be 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 [1].

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 modelling of goals and scenarios is complementary and may 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.

3 A Design Methodology Based on Goal and Scenario Modelling

To support early requirements engineering and high-level system design, our goal and scenario modelling methodology aims to elicit, refine and operationalize customer-specific requirements incrementally based on domain experts' knowledge, until a satisfactory design is found. In this process, the 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' viewpoint, and seeking for a balanced solution for all. Another advantage of GRL is that not only functional requirements but also non-functional requirements (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.

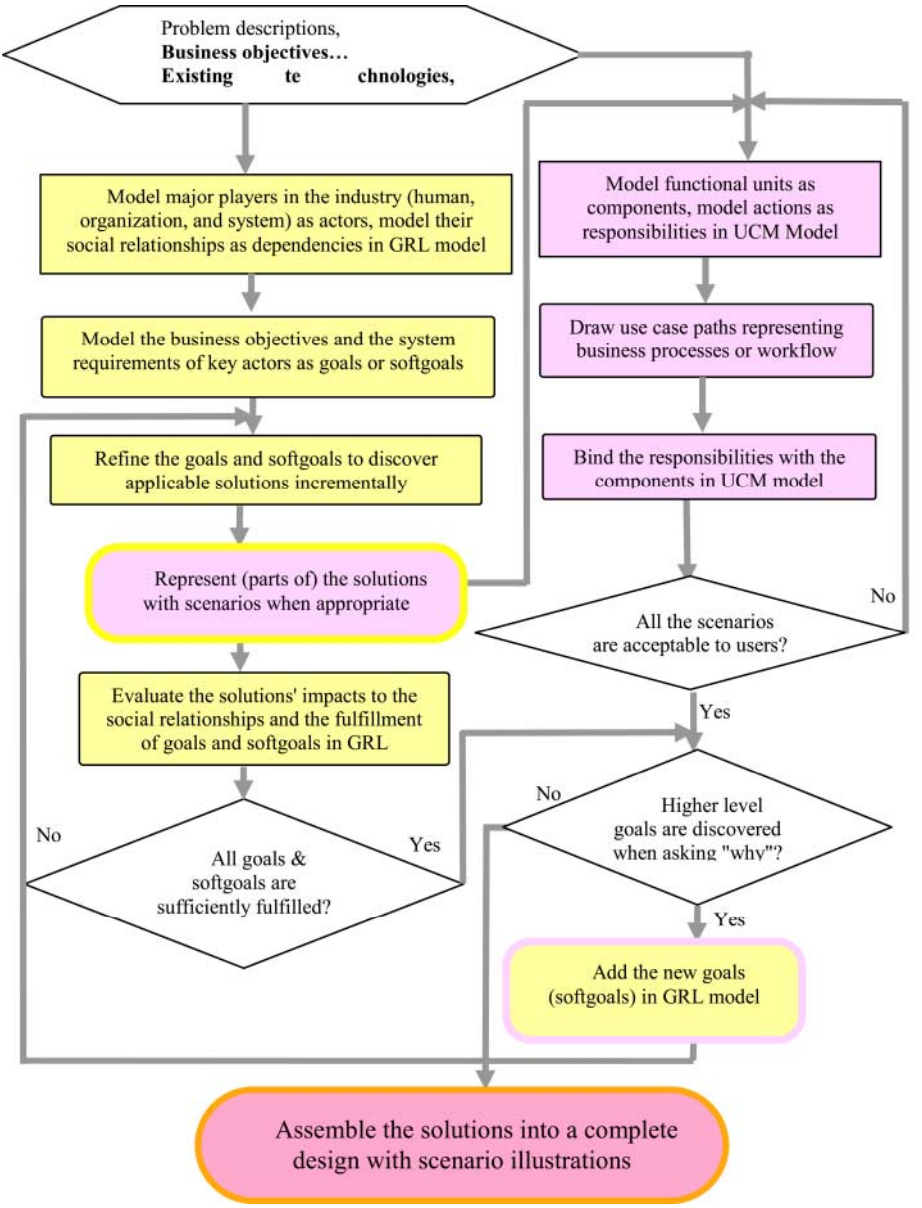

Fig. 2 Goal and scenario modelling based system design process

In comparison, it is often possible to create operational scenarios about using the hypothetical system relatively easily. In our approach, in parallel with goal-oriented modelling, UCM scenarios are used to describe the behavioral features and designs of

the intended system in some restricted usage contexts. The scenarios basically 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?" Then, by raising „why“ questions about these scenarios (e.g. „why let learners lead the class instead of instructor?“) some implicit system goals are made explicit.

The general steps of the process are illustrated in Figure 2. From the flow chart, 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, first actor dependency models are 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 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. Scenarios illustrating the business processes or workflow are also obtained.

4 Case Study: Designing a Web-Based Training System

To illustrate the complementary application of GRL and UCM, we use the example of designing a Web-Based Training (WBT) System [6]. The approach is applicable to information systems in general, where there are conflicting goals and tradeoffs during design. A case study in telecommunication domain is discussed in [10], which focuses more on using goal and scenario together in software architectural design. Starting from the identification of the major stakeholders of the domain, we explain in sequence how to capture the original business objectives of the stakeholders, refine and operationalize these objectives into applicable design alternatives with GRL and how to visualize and concretize some solutions with UCM.

Step 1: Placing system design within its broader social context [15] (as in Figure 3), the proposed modelling approach can help to address the following questions systematically: Who are the major players in the business domain? What kinds of relationships exist among them? What are the business objectives and criteria of success for these players? In Figure 3, circles denote actors in the domain. If there is a line above the actor’s name, the actor is an agent (or class of agent) with physical existence. Actors with a line under their name are abstract roles that can be played by agents.

The various dependency links in the model depict that in web-based training, the course provider is a key player, who provides web-based education service to learners. At the same time, he/she may depend on the support of web-technology expert, course content provider and web training consultant. 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 e-training domain, and is a reusable domain knowledge model.

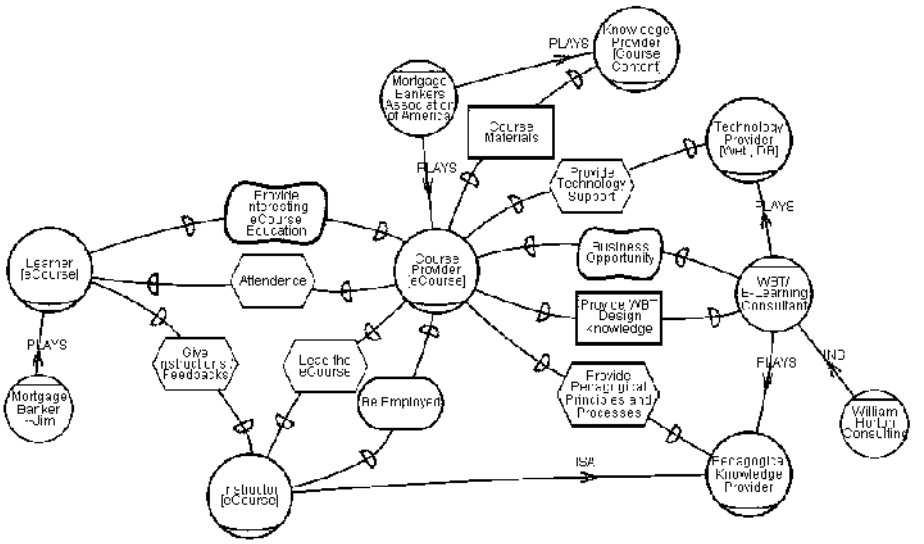

Fig. 3 Major players in E-Learning domain, agent dependency relationships, role-playing relationships and agent classification

Step 2: After the main vendors are identified, we ask them what are their business objectives, i.e., what they hope to accomplish for their organization, their sponsors, or their financial backers. Assume that, in our specific e-training system, the course provider is "Mortgage Bankers Association of America", who has two things in mind:

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

They are represented as two softgoals in the initial GRL goal model in Figure 4.

Step 3: Explore the alternative business processes, methods or technologies used in this industry or business. Evaluate how are these alternatives serving the specific business objectives and the quality expectations of stakeholders.

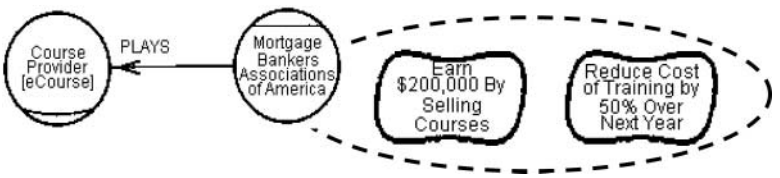

Fig. 4 Business objectives represented as softgoals in original goal model

In Figure 5, we see how the two solutions „Web-Based Training“ and „Conventional Classroom Training“ (represented as task nodes) contribute differently to the goals. By using contribution links labelled with numbers or different symbolic types, the model portrayed that WBT ***makes*** the goal of „Reduce Cost of Training by 50% Over Next Year“ satisfiable, while conventional training method ***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 analysis suggests that WBT may be a better option for current stakeholder. The upper part of this model (the two softgoals and the help relationship between them) is only applicable to current system, while the lower part (the structure showing the different resource consumption of the two solutions) depicts generic domain knowledge reusable to all course providers of web-based training system.

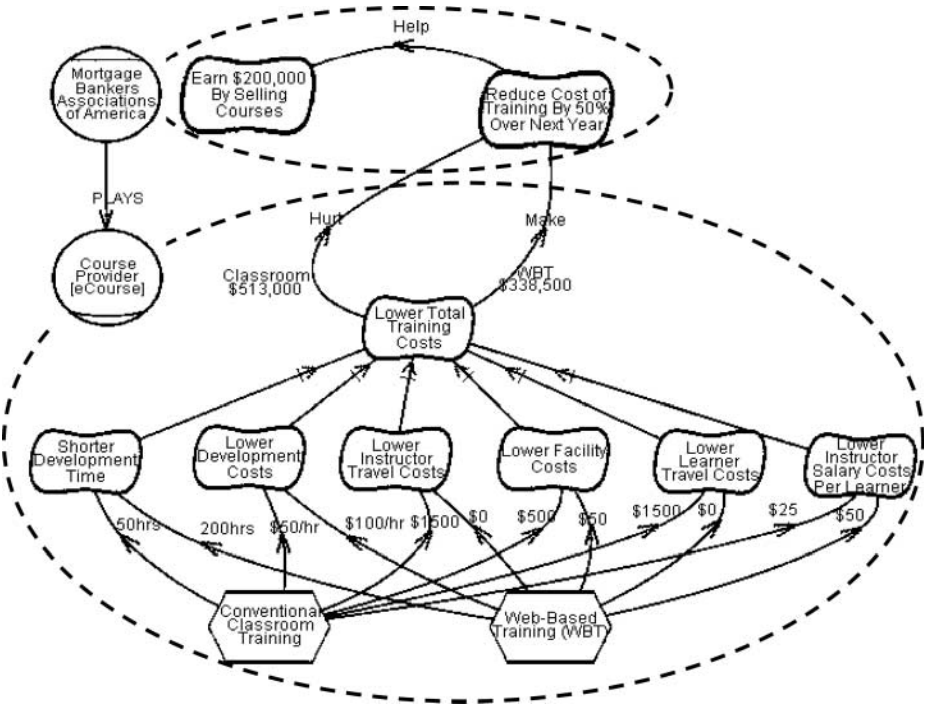

Fig. 5 Evaluate alternative technologies by comparison of resource consumption

Step 4: The advantages and disadvantages of the candidate solution are further investigated by evaluating its contributions to other concerned softgoals. For each disadvantage, mitigation plans are considered to complement the current solution.

The corresponding goal model (Figure 6) shows that the advantages of WBT include „Costs Saved“, „Better Teaching Techniques Enabled“, „Collaborative Learning Promoted“, and „Effective Learning Technologies Used“. Consequently, the overall „Quality of Learning Improved“. It also contributions positively to „Globalization“, „Flexibility“, both of which contribute positively (helps) to the learner's satisfactory, as the right hand side of the model suggests. On the left side, disadvantages are considered, e.g., the inherent „High Dropout Rates“ and „More Efforts“ on „Conversion“ and „Electronic Delivery“ of WBT hurts the high level goals of the stakeholder. These disadvantages can be mitigated by countermeasures such as „Require Commitment“, which are represented tasks connected with a negative correlation links (the dotted lines with arrows) to the unfavorable contributions links in the graph.

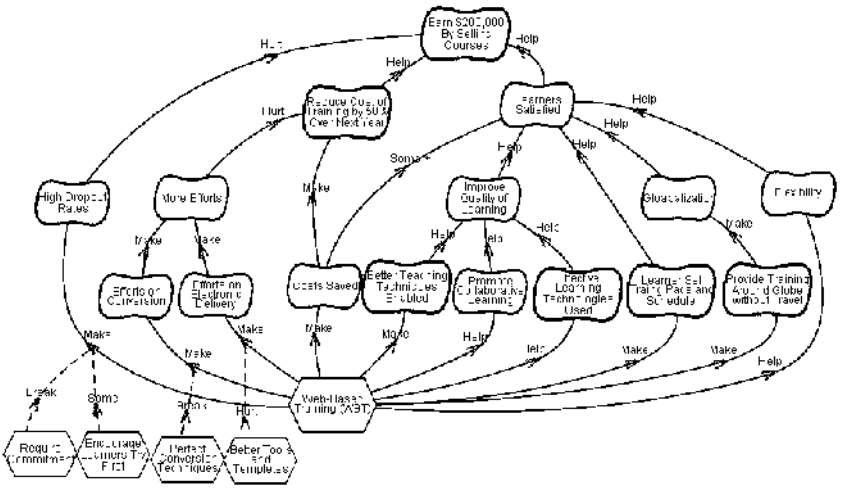

Fig. 6 Advantages, disadvantages and mitigation measures connected with contribution links

GRL evaluates the satisfaction of softgoal via a qualitative labeling procedure [3]. 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.

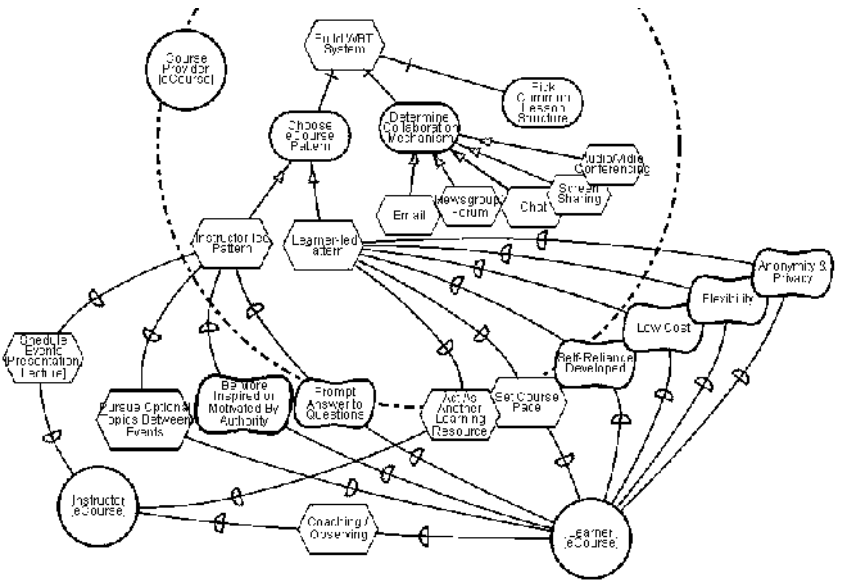

Fig. 7 Alternatives to implement essential sub-components and their impacts on actor dependencies

Step 5: Identify the alternative essential sub-processes/components to implement the candidate solution. The model in Figure 7 elaborates the generic knowledge about „Build a WBT System“. First of all, an e-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 subgoals connected to the root task with decomposition links.

Considering course patterns, the designers have two main options: instructor-led course and learner-led course. In Figure 7, the two task nodes are connected to parent goal with means-ends link. The dependency links pointing from these two tasks tell us that the two solutions lead to quite different role designations – the teacher is the driving force in instructor-led courses, but only acting as one of the optional learning resources in learner-led training. Conversely, the dependencies pointing to the two task nodes show that they have different capabilities and qualities to offer. Learner-led training has „Lower Costs“. Learners are more „Flexible“ on their schedule and learning content, and they also appreciate the „Anonymity and Privacy“. In instructor-led training, instructors can „Answer questions and solve problems promptly“ (as they arise), provide the authority needed by some learners for „Motivation“, and urge and „Inspire“ learners in a humanized way.

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 can work out an acceptable design.

Step 6: As the goal-oriented design proceeds, finer-grained analysis needs to be conducted, hence the scenario-based notation comes into use. To elaborate the goal „Pick Lesson Structure“ in Figure 7, alternative structures are denoted in the GRL model in Figure 8 as task nodes having different usage. For instance, „Classic Tutorials“ are good to teach basic knowledge and skills, as they provide a „Safe“, „Reliable“ and „Simple“ way.

On the right side of Figure 8, each of the first five class structures is described as a UCM scenario. WBT System and Learner, are represented as agent components (rectangles), holders of responsibilities (small crosses along on the wiggle lines). The first scenario means that, in a classic tutorial, after an introduction, learners read 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.

The second and fourth scenarios read similarly as the first one. In the third scenario, the use case path branches for different learners if they choose different subjects in the course. In the fifth scenario, the learner and the WBT system collaborates on searching on the web for materials the learner is interested in, so they are sharing responsibilities (denoted by adding a square „S“ between the shared responsibilities).

The last scenario uses GRL and UCM modelling constructors jointly, as neither GRL nor UCM can describe this case on their own. In this scenario, both static and dynamic features needs to be considered. Tasks and decomposition links are used to

describe the composition of the course materials, while the use case path is used to capture the dynamic generation of course structure at runtime. This model suggests that a close coupling of the two notations is needed for some applications.

In the case study above, the UCM model are rather simplistic because we have only tackled the highest level of process design, and the process in e-training is not very complicated. As we go down to a sufficiently detailed design, a UCM model may 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 counterpart to GRL in the process from requirements to high-level design, because it provides a concrete model of each design alternative. Based on the features in such a model, new non-functional requirements 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 always be explored and concretized in a UCM model. Thus the above design process may iterate several rounds until an acceptable design is made.

5 Related Work

The work of this paper builds on an original submission to ITU-T Study Group 10 on the topic of User Requirements Notation (URN) [13]. The User Requirements Notation (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. This standard shall specify functional requirements in UCM and non-functional requirements in GRL as well as a set of relationships between the GRL and UCM. The methodology introduced in this paper is a follow-up step in relating the two modelling notations.

The combined use of goals and scenarios has been explored within requirements engineering, primarily for eliciting, validating and documenting software requirements. Van Lamsweerde and Willement studied the use of scenarios for requirements elicitation and explored the process of inferring formal specifications of goals and requirements from scenario descriptions in [8]. Though they treat goal elaboration and scenario elaboration as intertwined processes, their work regarding scenarios in [8] mainly focuses 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. The fundamental point is that both the goal-oriented modelling in GRL and the scenario-based modelling in UCM run through requirements to design, and also their interactions.

In the CREWS project, Rolland et al. have proposed the coupling of goals and scenarios in requirements engineeing with CREWS-L'Ecritoire [11]. 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". Their work focuses mainly on the elicitation of functional requirements/goals. In GRL, both functional and non-functional requirements are considered, with special attention being paid to non-functional requirements. The modelling process involves both requirements engineering activities and high-level architectural and process design.

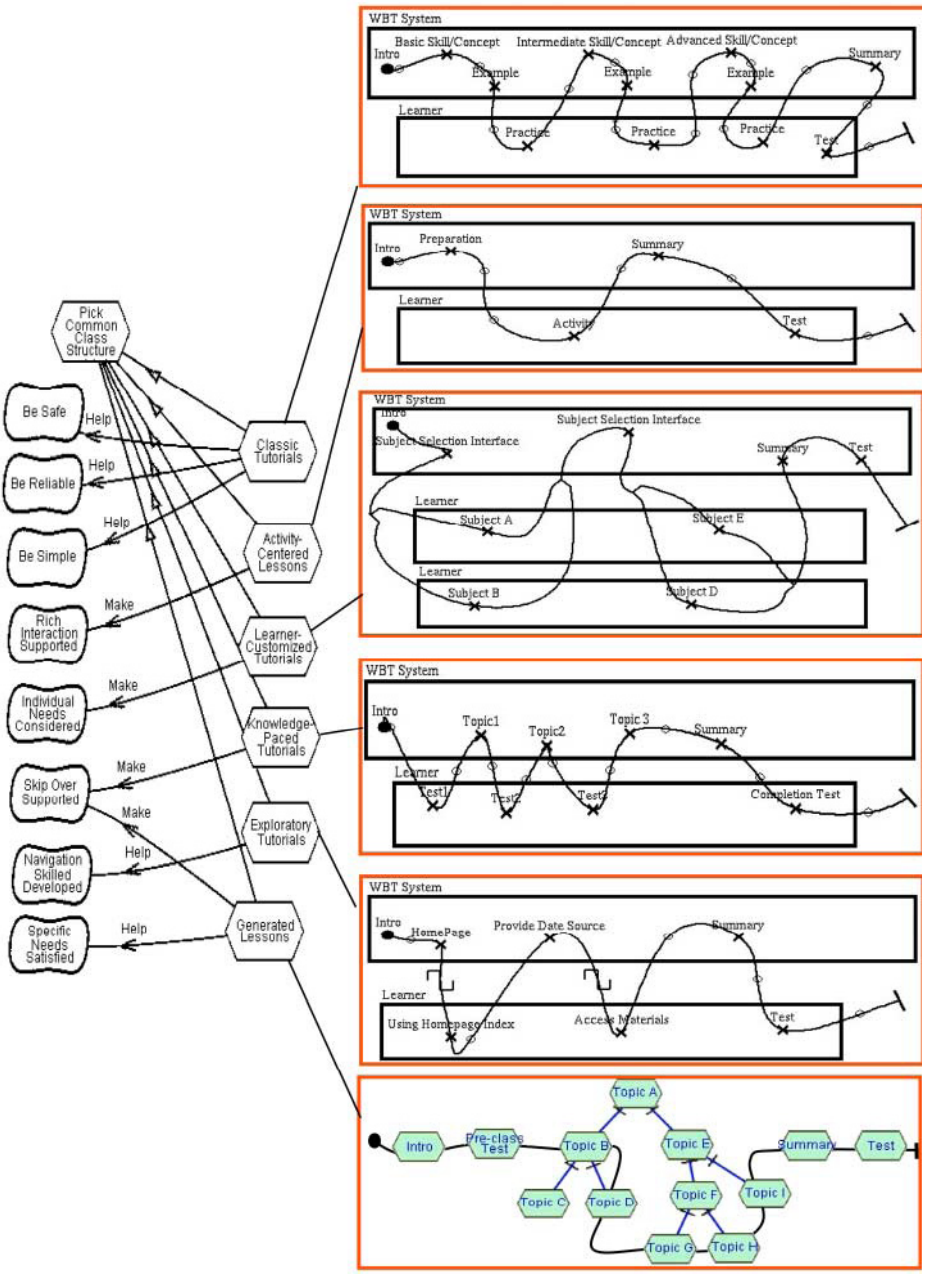

Fig. 8 Design alternatives represented as tasks, and their corresponding scenarios

The Software Architecture Analysis Method (SAAM) [7] 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 GRL+UCM, scenarios are more design-oriented, being concerned with refinement of system requirements. The quality of the architectures corresponding to these scenarios is judged based on expert knowledge as the design proceeds.

6 Conclusions and Future Work

Goals and scenarios complement each other not only in requirements engineering but also during the incremental system design process. The combined use of GRL and UCM enables the description of functional and non-functional requirements, abstract requirements and concrete system models, intentional strategic design rationales and non-intentional details of concurrent, temporal aspects of the future system.
The coupling of goals and scenarios in our current approach is loose, as goal and scenario models can be constructed separately. One scenario 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 scenario model 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. Tighter coupling may be investigated in the future to provide more guidance and support.

GRL and UCM are vehicles for expressing knowledge. To make better use of them, we need to acquire both software design knowledge and more knowledge of various domains, and represent this knowledge in GRL and UCM 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 scenarios can be embeded into the goal models and validated by using model-checking techniques [4].

Acknowledgements

The work of this paper is motivated by an original submission to ITU-T study group 10 on the topic of User Requirements Notation (URN). The kind cooperation of people from Mitel Networks, Nortel Networks and other institutions is gratefully acknowledged. This work received financial support from NSERC, CITO, and Mitel Networks.

References

1. Buhr, R. J. A. Use Case Maps as Architectural Entities for Complex Systems. In: Transactions on Software Engineering, IEEE, Vol. 24, No. 12, December 1998, pp. 1131-1155.
2. Carroll, J. M. Introduction: The Scenario Perspective on System Development. In Scenario-Based Design: Envisioning Work and Technology in System Development, Ed Caroll, J. M. 1995. pp. 1-17.
3. Chung, L., Nixon, B. A., Yu, E.and Mylopoulos, J. Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, 2000.
4. Fuxman, A., Pistore, M., Mylopoulos, J., and Traverso, P. Model Checking Early Requirements Specifications in Tropos. In Proceedings of the 5th IEEE International Symposium on Requirements Engineering. August 2001. Toronto, Canada. 174-181.
5. GRL web site. http://www.cs.toronto.edu/km/GRL/.
6. Horton, W. Designing Web-Based Training, John Wiley & Sons, 2000.
7. Kazman, R., Bass, L., Abowd, G. and Webb, M. SAAM: A Method for Analyzing the Properties of Software Architectures. In *Proceedings of the 16th International Conference on Software Engineering*. May 1994. Sorrento, Italy. 81-90.
8. Lamsweerde, A. V., Willemet, L. Inferring Declarative Requirements Specifications from Operational Scenarios. *IEEE Transactions on Software Engineering*, Special Issue on Scenario Management, December 1998.
9. Lamsweerde, A. V. Requirements Engineering in the Year 00: A Research Perspective. In the *Proceedings of 22nd International Conference on Software Engineering*. Limerick, June 2000, ACM press.
10. Liu, L., Yu, E. From Requirements to Architectural Design - Using Goals and Scenarios. 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/.
11. Rolland, C., Grosz, G. and Kla, R. Experience With Goal-Scenario Coupling In Requirements Engineering. In *Proceedings of the IEEE International Symposium on Requirements Engineering* 1998. June 1999. Limerick, Ireland.
12. Simon, A. H. The Sciences of the Artificial, Second Edition. Cambridge, MA: The MIT Press, 1981.
13. URN web site. http://www.usecasemaps.org/urn/.
14. Yu, E. and Mylopoulos, J. Why Goal-Oriented Requirements Engineering. In *Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality*. June 1998, Pisa, Italy. E. Dubois, A.L. Opdahl, K. Pohl, eds. Presses Universitaires de Namur, 1998. pp. 15-22.
15. Yu, E. 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.
16. Yu, E. Agent Orientation as a Modelling Paradigm. Wirtschaftsinformatik. 43(2) April 2001. pp. 123-132.