Using Non-Functional Requirements to Systematically Support Change*

Lawrence Chung
Computer Science Program
The University of Texas at Dallas
P. O. Box 830688, Richardson, TX 75083

Brian A. Nixon & Eric Yu
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4

Abstract

Non-Functional requirements (or quality requirements, NFRs) such as confidentiality, performance and timeliness are often crucial to a software system. Our NFR-Framework treats NFRs as goals to be achieved during the process of system development. Throughout the process, goals are decomposed, design tradeoffs are analysed, design decisions are rationalised, and goal achievement is evaluated. This paper shows how an historical record of the treatment of NFRs during the development process can also serve to systematically support evolution of the software system. We treat changes in terms of (i) adding or modifying NFRs, or changing their importance, and (ii) changes in design decisions or design rationale. This incremental approach is illustrated by a study of changes in banking policies at Barclays Bank.

1 Introduction

Non-Functional requirements (or *quality requirements, NFRs*) such as confidentiality, performance and timeliness are often crucial to a software system, such as a banking system. Furthermore, the continued success of a software system often requires meeting changing organisational needs in a competitive marketplace [23]. In fact, changing and conflicting requirements were one of the three main problems for software development, as identified in empirical studies [15]. But how can change be dealt with, given the need for quality (as embodied in NFRs)?

To systematically deal with change while aiming for high software quality, this paper adopts the NFR-Framework [9, 27], by treating quality requirements as goals to be achieved during the process of system development and change; throughout the process, goals are decomposed, design tradeoffs are analysed, design decisions are rationalised, and goals are evaluated. This process not only serves as a means for developing and improving quality software but also results in an history record. This paper shows how an historical record of the systematic treatment of quality requirements during the development process can also serve to support evolution of the software system.

Changes to existing software requires dealing with the impact of organisational changes, changes in specific requirements (e.g., increasing security for a particular operation) and workload (e.g., the number of bank clients). This involves identifying important changes, determining their potential impact on quality issues, and incorporating the impact through the process of improvement. The effect of changes can include a change in the target system selected, and a change in the degree of meeting requirements.

In terms of the historical record, dealing with changes in quality requirements of an existing software system amounts to (i) adding or modifying a quality requirement, or changing its importance, and (ii) extending the history of development and change while reflecting changes in the organisation, including its workload, and relating induced changes to appropriate components of the history record. This incremental approach needs a basis to record history, justify changes to the history record of the old software components, and rationalise the change process in relation to the old system and a trace of its sources.

Scenario. This process-oriented incremental approach is illustrated by a study of dealing with quality in a changing bank-loan system. We consider the impact of changes in requirements, primarily those driven by changes in organisational policy. The scenario is based on two editions of *The Barclays Code of Business Banking* [2, 3] which is in real-world use, setting out the general principles and terms of Barclays Bank's dealings with its business customers. From the Code we see that the bank has a number of quality concerns, including security, accuracy and timeliness. Furthermore, we assume that performance is also an important quality concern. The first edition [2] identifies a number of changes which were to be implemented in 1992, many of which relate to offering clients several options for more informative reporting on statements. This

*The first author was at the Department of Computer Science, University of Toronto when a draft of this paper was prepared. A longer version is available from the authors. Authors' e-mail addresses: `chung@utdallas.edu`, `{nixon,eric}@ai.toronto.edu`.

trend towards greater informativeness continued in the second edition [3], which is quite similar to [2]. We consider how our approach might be used to help Barclays deal with quality requirements, which have an impact on goal achievement and system design.

Since the NFR-Framework supports the requirements engineering process with components (goals, decisions, rationale, etc.) already included in the historical record, our use of the framework for dealing with change is facilitated. We treat both the sources and effects of change in the historical record in terms of changes in the components.

This paper extends the usage of the NFR-Framework [27, 9] by using its facilities to deal with *change.* This is illustrated by an initial study of dealing with change in a bank loan system. This draws on our studies of attaining quality for banking systems [31, 9, 10, 11]. The combination of performance and requirements for accuracy, timeliness and informativeness, is also treated in more detail than in [11]. We also offer guidelines for consistently managing historical records in the presence of changes in the world.

In Section 2 the NFR-Framework is illustrated in describing how we use the bank's initial requirements to generate an initial design, which is changed in Section 3 to deal with changes in requirements, based on trends in the Barclays Codes. Section 4 describes guidelines for achieving desirable properties that an historical goal record should possess. Section 5 discusses related work, and Section 6 summarises the paper.

2 Generating the Initial Design

2.1 Initial Top-Level Requirements

We first consider a development process which reflects the situation prior to the 1992 changes in the Barclays Code. A number of quality requirements are dealt with.

In reviewing Barclays' policies, we identified important *NFR goals,* some of which are captured and shown in the *goal graph* of Figure 1 (adapted from [11]). The nodes represent goals, which stand for requirements. Each goal has a *sort,* e.g., Security, which indicates the kind of requirement associated with the goal, and a *parameter,* e.g,. [loan-info], to indicate the subject of the goal. This goal means that loan information should be processed securely. Similarly, the other top goals mean that loan information should be informative, processed by the system with good time performance, i.e., rapidly, and be maintained securely.

Decompositions are one type of *method* for refining a goal. For example, the parent goal Time[loan-info] can be decomposed into *offspring* goals Time[change Base Rate] and Time[produce Statements], which are connected by an *AND* link. The interpretation is that *if* the base rate can be changed quickly, *and* statements can be produced quickly, *then* we have some confidence that fast processing of loan information can be "satisficed."[1] Besides *AND* links, other *link types* (beyond [29]) are available, which indicate different relationships between parent goals and their offspring.

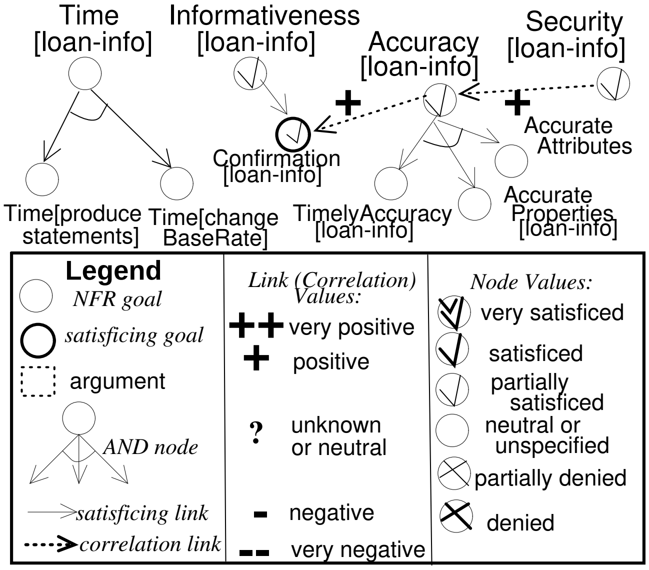

Figure 1: Initial top-level goal graph for loan system.

The use of the Framework requires the capture of expert knowledge about domains and NFRs. For example, we turn to security experts [19] to see that meeting an accuracy goal will contribute positively to meeting a security goal; this is shown as a positive *correlation link* (dotted line in figure). Furthermore, the experts state that confidentiality, availability and accuracy all contribute to security, and this is an example of a generic *decomposition method.* Similarly, accuracy can be decomposed [9] into goals for accurate attributes, accurate properties, and timely accuracy (Section 3).

In moving towards alternatives for target systems, *satisficing goals* (dark circles in figures) are intended to satisfice parent NFR goals. Barclays policies state that loan information be confirmed in writing for informativeness. Selecting the satisficing goal of *confirmation* contributes not only to meeting (shown with "$\surd$" in figure) informativeness but also accuracy, which in turn helps security. In order to support the development process, we have catalogued [7, 10, 9, 30, 31, 32] in a knowledge base such expert knowledge about sorts, decompositions, correlations

[1] Since goals representing NFRs are rarely "satisfied" in a clear-cut sense, but decisions do contribute to, or hinder, a particular goal, we use goal *satisficing* [35] when software is expected to satisfy NFRs within acceptable limits.

and satisficing goals, for the NFRs of accuracy, security and performance.

2.2 The Initial Design

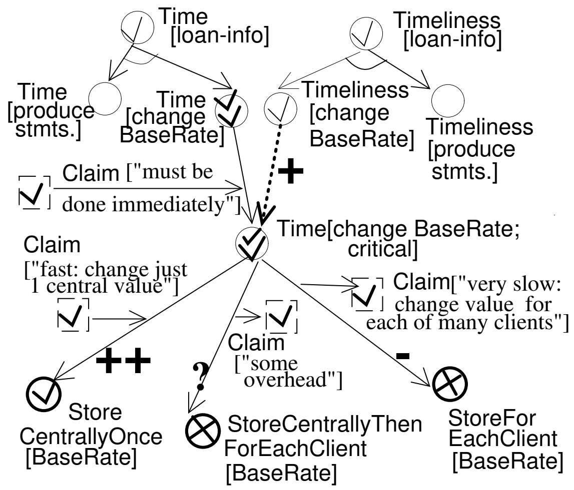

Figure 2: Goal graph for partial initial design.

To generate an initial design (Figure 2), we continue dealing with the initial quality requirements, focussing on time performance (from Figure 1) and timeliness. The bank can change interest rates (This is not a change in NFRs) during the loan period, and wants to keep clients informed in a timely manner. Interest rates on loans have two components: a *base rate* (prime rate), which is set nationally by the bank for all customers, and an *interest margin* which is set individually for each customer based on the risk involved. The bank gives advance written notice of changes in the interest margins. Changes in the base rate take effect immediately for all clients, and are advertised in the national press, without advance notice. We consider a change in base rate prior to 1992. (This discussion also applies after 1992, but it will interact with changes in requirements (Section 3).)

The need to identify and focus on *critical* goals has been pointed out for quality goals [24] including performance goals [36]. Thus we identify the time performance goal for changing the base rate as critical; we support this by attaching an argument (Claim), here based on organisational policy, that base rate changes must be done immediately. Now *timeliness,* the provision of information in a timely way, will be helped by the system quickly changing the base rate, since such a system can help notify customers of the change in a timely manner. This positive correlation link is detected and shown in the figure with a "+" sign.

In moving towards a target system, one alternative is to store the new rate for each client. This is ruled out by arguing, based on *workload* statistics [4, 5] that it is very slow (shown as "-" in Figure 2), since each of the many client records must be updated.[2]

Another alternative is to store the base rate in one central value in the system. As only one value must be changed, this would provide a very fast method (shown as "++") of bringing the new base rate into effect, hence is chosen for the target system. A third alternative is an hybrid of the other two methods, first changing the centrally stored rate, then updating the rates stored in all client records. Due to overhead, it offers intermediate ("**?**") time performance and is not selected now, but will be reconsidered in Section 3.

Now we have labelled some leaf nodes as satisficed ("$\surd$") or denied ("$\times$"). To determine the impact on higher-level goal nodes, the labelling algorithm is used. Working bottom-up, it takes into account the labels of offspring, and the link type (e.g., "+", "-"). For example, storing the base rate once, when combined with its very positive ("++") parent-offspring link, leads to good satisfaction of the critical time goal. This in turn has some positive contribution to the timeliness goal for changing the base rate, and to the overall goals for time performance and timeliness (top of Figure 2). A satisficed offspring, when combined with a *negative* link, leads to a denied parent; for example, if we had chosen StoreForEachClient, the critical time goal would have been denied.

3 Changing the Design to Deal with Changed Requirements

3.1 Changing the Design

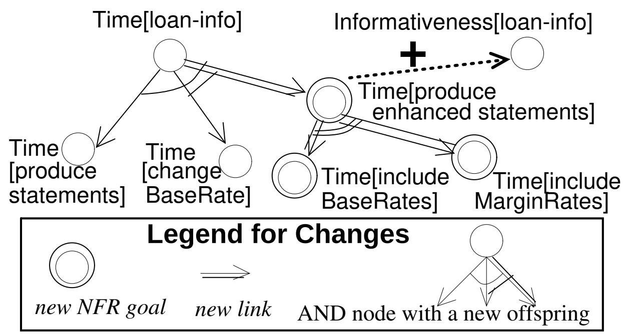

Figure 3: Goal graph with new requirement.

Offering loan clients more options for reporting on their statements entails a change in requirements. We consider

[2] While we do not have the number of clients available to us, in 1993, the average balance of loans and advances to customers of United Kingdom offices was 60 230 million Pounds Sterling, and the worldwide staff numbered 97 800.

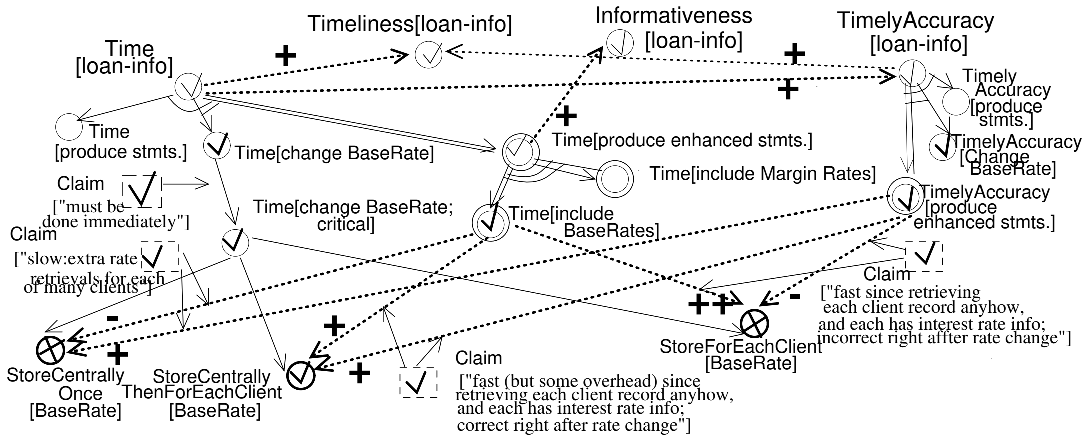

Figure 4: New design with a new requirement.

adding a new requirement, for good time performance for producing enhanced statements. We assume that enhanced statements include a list of the base rates and interest margins applicable for the statement period.

How do we *add* the requirement? Figure 3 shows the new time requirement which is in turn decomposed into time goals for including both base rates and margin rates in statements. The figure's legend shows how new goals and links are presented. In Figure 4, the new requirements result in *interaction* with existing requirements, leading to new tradeoffs and a new target system, with new arguments leading to the revised system. In particular, there is interaction between changing the base rate (Figure 3) and producing statements. This leads to a tradeoff between timely accuracy and time performance.

Figure 4 also shows a goal for Timely Accuracy (the timely recording of information in the system) of loan information. This would be decomposed, prior to 1992, into goals for accuracy for changing the base rate, and for producing (basic) statements. After 1992, another offspring, for the accuracy of producing enhanced statements, would be added. A similar decomposition was shown previously in Figure 3, for a time performance goal.

Now consider the performance impact of including the base rates upon the alternatives for a target system, and the associated arguments (This is an example of "downward correlation" [9]). Storing the base rate for each client will result in very fast time performance (shown as "++"), since preparation of each statement must involve retrieval of each client record (containing the base rates) anyhow. Due to overhead, reasonable time performance, but a little slower ("+"), would result from the hybrid method of storing the rate centrally, than for each client. Storing rates centrally would result in poor performance (shown as "-") due to the extra rate retrievals required for *each* client; we can argue, based on the organisation's *workload*, that this choice is very poor due to the large number of bank clients.

The choice of target system design is not yet complete, however, due to the impact of the target alternatives upon the Timely Accuracy goal for producing enhanced statements (an example of "upward correlation" [9]). Here an accuracy issue stems from including the base rate in statements. Suppose the base rate is changed just before statements are prepared. Now if only one base-rate value is changed in the entire system, this can be quickly changed, preserving accuracy. However, if the rate is changed for each client, this might take a long period of time to complete, and some statements could incorrectly show only the old base rate. The hybrid method can maintain accuracy, as it checks whether the base rate has *recently* been changed before retrieving a value: if there has been a recent change, it uses the (correct) central value; otherwise it uses the (correct) value in the client record. Thus the arguments show that in producing enhanced statements, there are tradeoffs between time performance and timely accuracy. As accuracy is a very important corporate goal, it is given priority, and storing the base rate for each client is ruled out, as it sometimes gives incorrect results. On the other hand, performance is also important, so storing the rate centrally alone (the prior choice) is ruled out, as it is generally slow, even though accurate. Thus in the presence of the accuracy-performance tradeoff, the hybrid method (ruled out with the previous requirements, due to overhead) is now chosen as it offers timely accuracy and also reasonable time performance for both changing the base rate and including the rate in statements. Thus an organisational change has resulted in a change in the target system.

Given the choice of target system, the framework's labelling procedure shows the impact on quality requirements. Here, for example, the choice of the hybrid method has a positive impact upon timely accuracy of producing enhanced statements, and upon time goals for both changing the base rates and including them in statements. This last improvement has a positive impact on meeting the time performance goal for producing enhanced statements.

3.2 Overall Impact of Change

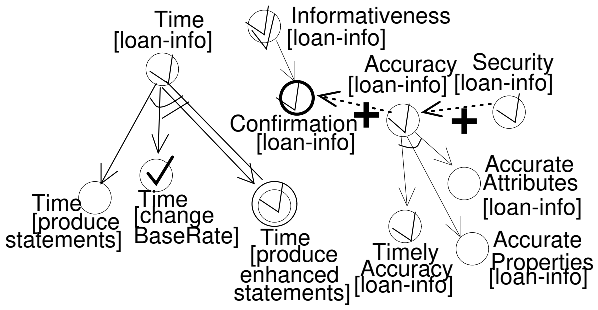

Figure 5: Impact of new requirement on top-level goals.

What has been the overall impact of the change in requirements on (other) top-level requirements?

1. *The goal graph (Figure 5) now includes the history of dealing with the change.* Goals and offspring goals have been added to reflect changed requirements, together with the interactions among the initial and additional goals. Tradeoffs were considered and arguments were revised.

2. *This has lead to a change in the target system.* This was driven by changes in requirements, and was rationalised by changes in arguments.

3. *This has also changed the extent to which we have satisficed the overall top-level requirements.* Here, the effect was to satisfice the new goals (e.g., TimelyAccuracy[produce enhanced statements]), which contributed *positively* towards the top-level goals (e.g., TimelyAccuracy[loan-info]). As part of the tradeoff, however, time performance for changing the base rate went from being very satisficed to being satisficed. Comparison of Figure 5 with Figures 1 and 2 shows the overall impact: *an increase in informativeness and timely accuracy, hence timeliness, with some time performance cost* due to the extended processing needed for producing the enhanced statements.

To simplify the presentation, some nodes (both those before and after the changes) were not decomposed to satisficing goals, or were not labelled. So here, without satisficing the offspring (here, of *AND* nodes) we cannot label their top-level parents (e.g., Time[loan-info]) as satisficed. What we can say about such unlabelled goals is that if the initial goals were satisficed and remained satisficed, they would contribute to satisficing the parents. Likewise, satisficing new offspring will help to satisfice the parents.

4 Guidelines for Changes

Dealing with changes involves identifying changes and associating them with appropriate components of the history record, i.e., goal graph. This process is often interleaved, and proceeds incrementally. Tools should provide semi-automated support of a developer dealing with change. We have developed tools for using the basic NFR-Framework,which primarily help the developer state and refine goals. But now, dealing with change may require operations, such as removal of a goal, which temporarily make the goal graph inconsistent. In order to help restore the goal graph to a consistent state, we offer some initial guidelines, in terms of some syntactic and semantic structural consistency criteria.

4.1 Desirable Properties of Goal Graphs

The process of building and revising a goal graph is facilitated by a number of notions.

The use of the notion of *structurally traceable goal graph* provides some "syntactic" principles for maintaining the structure of a goal graph:

- Each argument should be attached to a goal or a link.
- Each link should have its direction (positive, negative or neutral) and strength (strong, weak, to-be-specified) of contribution specified.

These guidelines help the labelling procedure to determine the whether goals are satisficed.

There are other notions in a longer version of this paper (available from the authors). These notions help trace decisions to identifiable sources of knowledge, help trace the relationships among design decisions, their corresponding goals, and the people who stated the goals, and help ensure that changes are rationalised, evaluated, and compared to the previous system.

4.2 Guidelines for Categories of Changes

Incorporating revisions into a goal graph is facilitated by division of NFR-related changes into three categories (corresponding to the three types of goals in the legend of Figure 1) and uses a set of guidelines for propagating the changes. The guidelines are based on these categories, the kinds of operations on quality requirements, and the structure of the goal graph.

Guidelines for Changes in NFR Goals. A change in a NFR goal usually proceeds in a downward direction, linking and expanding the goal to other NFR, satisficing and argument goals:

- *addition of a quality requirement:* A new goal should generally be linked with existing goals (NFR, satisficing and argumentation goals). If afterwards there remain any isolated NFR goals without offspring, we see if

offspring can be created and related to other existing goals. This involves an iterative stage of decomposing the added goals, considering design trade-offs and selecting satisficing goals, and providing arguments for or against design decisions.
- *deletion of a quality requirement:* This can involve recursively deleting all (incoming and outgoing) links associated with the requirement, or re-associating the links with another goal. A parent-less satisficing goal should be deleted or linked to an NFR goal. A parent-less argument usually needs to be deleted.
- *changing the importance (criticality) of a quality requirement:* This can be done by changing the criticality attribute of a relevant goal. This affects the way conflicts are resolved and the number and type of methods to be selected as well as their associated arguments.

Guidelines for Changes in Satisficing Goals. A change in a satisficing goal proceeds both upwards to parents, propagating it to satisficing and NFR goals, and downwards to offspring, propagating it to other satisficing goals and to argument goals. Although omitted due to space limits, there are guidelines for the cases of *addition, deletion* and *change in criticality.* Interestingly, addition may lead to new correlation links.

Guidelines for Changes in Arguments. Again there are guidelines for of *addition, deletion* and *change in criticality.* The guidelines each have two cases, depending on whether the changed argument *supports* or *denies* another goal. In addition, changes in criticality are sub-divided into strengthening and weaking of criticality.

5 Related Work

Related work includes work in decision support systems (e.g., [25, 33]) which influenced the argumentation aspects of the NFR-Framework. The current work adopts the NFR-Framework, hence using the ideas of design rationale. However, the current work is distinct as it deals with changes, and utilises semantic structures, such as link types, correlations and method instances, in guidelines for incorporating changes. The current work also demonstrates how such structures help annotation of changes.

A large empirical study of requirements *traceability* is reported in [17] which involves developing and following the life of a requirement from its origin through all phases of development. This approach is similar to ours concerning defects. However, we have looked at traceability with more emphasis on incorporating changes in NFRs, systematically detecting defects and supporting the process of corresponding changes in designs and implementations.

The improvement paradigm of Goal-Question-Metric in the TAME project [6] is similar to our approach in the sense that it uses the notion of goals and provides a way of argumentation via questions. TAME can be viewed as somewhat complementary as its goals are general requirements and it uses a collection of metrics for improvement, while we use NFRs as goals to systematically support improvement by way of decomposition, correlation and achievement, which are all integrated into a development history record serving as the basis for change. Metrics [13] can also play an important role in maintenance, to which our work is related concerning software evolution and traceability.

Work on requirements analysis for safety-critical systems [14] shares a similar spirit with ours in that both are concerned with NFRs and their analysis. Coombes and McDermid's work has taken a causal reasoning approach, hence is complementary to our amalgamation of qualitative reasoning from truth maintenance systems [1] and dialectical reasoning from work on design rationale [25]. The two are also complementary as [14] focusses on safety and its analysis, while our approach is directed to systematically supporting changes using NFRs.

This work grows out of the earlier DAIDA environment [21] which provides support for all phases of information system engineering. Our work extends the environment with principles and guidelines for using NFRs to guide evolution process, and our tool's use of DAIDA's knowledge base management facilities [22]. Our framework follows a decision-oriented (as opposed to activity- or product-oriented) approach to managing software evolution [20, 34], but provides additional representation and support, and is driven by NFRs.

6 Conclusions

Observations. In dealing with change in this study, we observe that our operations on the goal graphs has mainly involved: adding a new goal (e.g., due to changes in organisational policies); revising an existing interaction (e.g., correlations); considering new interactions; (re-) considering alternative target systems; and providing arguments for the chosen system, e.g., based on the organisation's workload, such as the number of clients. In our experience this process was quite rapid: there was little additional material to add to the graph; and of the new material, some of the bookkeeping (e.g., some correlations) can be handled automatically. This kind of saving helps the developer to focus on the domain and the quality requirements. We feel that a main reason for this saving is that the NFR-Framework captures underlying structure and real-world knowledge (Cf. [18]), which is needed, used, and re-used.

Tool Support. To help a developer use the basic NFR-Framework, a tool, the NFR-Assistant, has been implemented to deal with accuracy and security [9, 10], and work is in progress on performance [32]. The tool helps the developer catalogue, browse and use knowledge about the domain, particular NFRs and their associated methods and correlations. To also free the developer from handling routine bookkeeping and simple reasoning, the tool, given a method to apply, generates offspring, helps detect correlations, and propagates changes in label values throughout the goal graph. However, this is very much a semi-automatic approach. It is up to *developers* using the tool to direct the overall development process, select focus, choose or supplement pre-defined methods, provide rationale, and use their own expertise about the domain and the NFRs. In this sense, we do not yet offer active guidance or a model for process enactment.

In future, the tool should also distinguish new goals and links from old ones, in extracting the differences between the old and new goal graphs. The tool should also be extended to support our guidelines, first by detecting violations of the properties of Section 4.1, and then by informing the developer of goals, links and labels which should be adjusted according to the guidelines of Section 4.2.

Future Work. Interestingly, this study has presented some support for extensions to the NFR-Framework, particularly in representing changes in strengths of goal achievement. Adding additional "shadings" to the *existing* values (e.g., satisficed, denied, neutral, etc.) would help us represent, for example, situations where a requirement is initially satisficed, and subsequently also satisficed, but to a greater degree. This could be viewed as adding some features of quantitative frameworks, while, at a high level of abstraction, retaining the features of a qualitative framework. The framework could also provide a convenient notation for reflecting changes made over time; a corresponding graphical notation would be helpful to the tools for display purposes.

Given the vast amount of informal, loosely-connected descriptions, we feel from this study the need for more structure to argumentation, perhaps by way of a model of organisation with a rich ontology, e.g., [37, 38].

Further studies of real systems could be made, to illustrate more of the cases of the guidelines presented in Section 4. The guidelines are organised by the structure of a goal graph. When the guidelines are suitably formalised, we could envisage demonstrations of correctness (by structural induction on the goal graph) and a formal verification that the labelling algorithm generates a unique and consistent assignment of labels. Studies would also help determine how easily the framework and its notation can be learned and applied by practitioners, as raised by experts in other domains (medical and governmental computing) who have reviewed our previous studies [12].

Conclusion. In dealing with change, our overall long-term goals are to increase quality, decrease development time and cost, and to improve conciseness of representation and schemes for assisting reasoning. We have taken an existing framework [9, 27] for dealing with quality requirements, and have provided a set of guidelines stating the mechanics of incrementally dealing with changes in the context of the framework. Our study of change in a banking environment shows that the framework's history structure provides a concise historical record of dealing with change.

Our approach leads to the following benefits:

1. *Incorporating change is conceptually simple,* since the same structure is used for developing both the initial goal graph and its subsequent revisions;

2. *Identification and treatment of incremental change is faster and cheaper,* since the initial goal graph can be consulted to detect patterns and reduce searches of documentation whose references to quality have already been captured as goals, methods, and interactions among old and new goals.

3. *Changes in NFRs are representable,* since a knowledge-based approach enables capturing concepts central to dealing with NFRs, both initial and subsequent ones.

The NFR-Framework is good for dealing with change, because it inherently deals with alternatives in a goal-oriented way. When there is change, some other alternative becomes appropriate because of the new circumstances. Managing change in terms of the means-ends (alternatives-goals) linkages is an important dimension because the underlying interactions are captured. The idea of applying means-ends reasoning to deal with change is also the basis for a framework for process reengineering [38, 39, 40]. Most work on change and evolution deals with the historical or temporal dimension, e.g., versioning and configuration management. It would be interesting to see how the means-ends reasoning dimension interacts with other dimensions in dealing with change.

Acknowledgments. Our thanks to Anthony Finkelstein, Stephen Fickas and Martin Feather for providing a copy of the Barclays Code of Business Banking for use as an example at the Workshop on Research Issues in the Intersection between Software Engineering and Artificial Intelligence, Sorrento, Italy, May 1994. Our sincere gratitude for all his help to Tim Baxter of Barclays Bank in London who provided us with the current edition of the Code. We are very grateful to Professor John Mylopoulos for his ongoing help in this work, and the referees for their excellent comments.

References

[1] *Artificial Intelligence Journal,* Special Issue on Qualitative Reasoning, vol. 24, nos. 1–3, Dec., 1984.

[2] Barclays Bank PLC, *The Barclays Code of Business Banking.* London, England, effective 31st Jan. 1992.

[3] Barclays Bank PLC, *The Barclays Code of Business Banking.* London, England, May 1993.

[4] Barclays Bank PLC, *Annual Review & Summary Financial Statement,* London, England, 1993.

[5] Barclays Bank PLC, *Report and Accounts,* London, 1993.

[6] V. R. Basili and H. D. Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments," *IEEE TSE,* vol. 14, June 1988.

[7] L. Chung, "Representation and Utilization of Non-Functional Requirements for Information System Design." In *Advanced Information Systems Eng.,* Proc., 3rd Int. Conf. CAiSE '91, Trondheim, Norway. Springer-Verlag, 1991.

[8] K. L. Chung, P. Katalagarianos, M. Marakakis, M. Mertikas, J. Mylopoulos, Y. Vassiliou, "From Information System Requirements to Designs: A Mapping Framework." *Information Systems,* Vol. 16, 1991.

[9] K. L. Chung, *Representing and Using Non-Functional Requirements: A Process-Oriented Approach.* Ph.D. Thesis, Dept. of Computer Science, Univ. of Toronto, June 1993. Also Technical Report DKBS–TR–93–1.

[10] L. Chung, "Dealing With Security Requirements During the Development of Information Systems." In *Advanced Information Systems Eng.,* Proc., 5th Int. Conf. CAiSE '93, Paris, France. Berlin: Springer-Verlag, 1993, pp. 234–251.

[11] L. Chung, B. A. Nixon and E. Yu, "Using Quality Requirements to Drive Software Development." *Workshop on Research Issues in the Intersection Between Software Eng. and Artificial Intelligence,* Sorrento, Italy, May 1994. Also in slightly revised form in: "Using Quality Requirements to Systematically Develop Quality Software." *Proc. 4th Int. Conf. on Software Quality,* McLean, VA, U.S.A., Oct 1994.

[12] L. Chung, B. A. Nixon, "Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach." To appear in *Proc., 17th ICSE,* Seattle, WA, U.S.A., Apr. 1995.

[13] D. Coleman, D. Ash, B. Lowther and P. Oman, "Using Metrics to Evaluate Software System Maintainability." *IEEE Computer,* vol. 27, no. 8, Aug. 1994.

[14] A. C. Coombes and J. A. McDermid, "Using Qualitative Physics in Requirements Specification of Safety Critical Systems — A Potential AI User's Perspective." *Workshop on Research Issues in the Intersection Between Software Eng. and Artificial Intelligence,* Sorrento, Italy, May 1994.

[15] B. Curtis, H. Krasner and N. Iscoe, "A Field Study of the Software Design Process for Large Systems." *Comm. of the ACM,* vol. 31, no. 11, Nov. 1988.

[16] A. C. W. Finkelstein and S. J. M. Green, *Goal-oriented Requirements Engineering.* Tech. Rept TR–93–42, Imperial College (London Univ.), forthcoming.

[17] O. C. Z. Gotel and A. C. W. Finkelstein, An Analysis of the Requirements Traceability Problem. *Proc. Int. Conf. on Requirements Eng.* Colorado Springs, 1994.

[18] S. Greenspan, J. Mylopoulos and A. Borgida, "On Formal Requirements Modeling Languages: RML Revisited." *Proc. 16th ICSE,* Sorrento, Italy, May 1994.

[19] European Communities, *Information Technology Security Evaluation Criteria, Provisional Harmonised Criteria,* Version 1.2, June 1991, Luxembourg.

[20] M. Jarke and T. Rose, "Managing Knowledge about Information System Evolution." *Proc. SIGMOD '88,* Chicago.

[21] M. Jarke, J. Mylopoulos, J. W. Schmidt, Y. Vassiliou, "DAIDA: An Environment for Evolving Information Systems," *ACM TOIS*, vol. 10, no. 1, Jan. 1992, pp. 1–50.

[22] M. Jarke (Ed.), *ConceptBase V3.1 User Manual.* Univ. of Passau, 1992.

[23] M. Jarke, K. Pohl, "Requirements Engineering in the Year 2001: On (Virtually) Managing a Changing Reality," *Workshop on System Requirements: Analysis, Management, and Exploitation*, Schloß Dagstuhl, Germany, 1994.

[24] J. M. Juran, F. M. Gryna Jr., and R. S. Bingham Jr. (Eds.), *Quality Control Handbook,* 3rd Ed., New York: McGraw-Hill, 1979.

[25] J. Lee, Extending the Potts and Bruns Model for Recording Design Rationale. *Proc. 13th Int. Conf. on Software Eng.,* Austin, May 1991, pp. 114–125.

[26] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis, Telos: Representing Knowledge about Information Systems, *ACM TOIS*, vol. 8, Oct. 1990, pp. 325–362.

[27] J. Mylopoulos, L. Chung, B. Nixon, "Representing and Using Non-Functional Requirements: A Process-Oriented Approach." *IEEE TSE,* Vol. 18, June 1992.

[28] J. Mylopoulos, L. Chung, E. Yu and B. Nixon, *Requirements Engineering 1993: Selected Papers.* Tech. Rept. DKBS–TR–93–2, Dept. Computer Science, Univ. of Toronto, 1993.

[29] N. Nilsson, *Problem-Solving Methods in Artificial Intelligence.* New York, McGraw-Hill, 1971.

[30] B. Nixon, "Implementation of Information System Design Specifications: A Performance Perspective." In *Database Programming Languages: Bulk Types & Persistent Data - 3rd Int. Workshop.* Morgan Kaufmann, 1992.

[31] B. A. Nixon, "Dealing with Performance Requirements During the Development of Information Systems." *Proc. IEEE Int. Symp. on Requirements Eng.,* San Diego, CA, Jan. 1993.

[32] B. A. Nixon, "Representing and Using Performance Requirements During the Development of Information Systems." In *Advances in Database Technology - EDBT '94,* Proc. 4th Int. Conf. Extending Database Technology, Cambridge, U.K. Springer-Verlag, 1994.

[33] C. Potts and G. Bruns, Recording the Reasons for Design Decisions, *Proc. 10th ICSE,* 1988.

[34] C. Rolland "Modeling the evolution of artifacts." *Proc. Int. Conf. on Requirements Engineering,* Colorado Springs, Colorado, U.S.A., Apr. 1994.

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

[36] C. U. Smith, *Performance Engineering of Software Systems.* Reading, MA: Addison-Wesley, 1990.

[37] E. Yu, Modelling Organizations for Information Systems Requirements Engineering. *Proc. IEEE Int. Symp. Requirements Eng.,* San Diego, CA, Jan. 1993.

[38] E. Yu and J. Mylopoulos, "Understanding 'Why' in Software Process Modelling, Analysis, and Design." *Proc. 16th ICSE,* Sorrento, Italy, May 1994.

[39] E. Yu and J. Mylopoulos, "From E-R to 'A-R' - Modelling Strategic Actor Relationships for Business Process Reengineering." *Proc. 13th Int. Conf. Entity-Relationship Approach,* Manchester, U.K., 1994.

[40] E. Yu, *Modelling Strategic Relationships for Process Reengineering.* Ph.D. Thesis, Dept. of Computer Science, Univ. of Toronto, 1994.