1 September 1994.
Submitted to RE '95,
The Second IEEE International Symposium on Requirements Engineering,
27–29 March 1995, York, England.

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 75083–0688, U.S.A.
chung@utdallas.edu
(214) 690–2178
Facsimile (214) 690–2349

Brian A. Nixon & Eric Yu

Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 1A4
{nixon,eric}@ai.toronto.edu
(416) 978–4299, 978–6027
Facsimile (416) 978–1455

Abstract

Quality requirements (or Non-Functional Requirements, NFRs) such as confidentiality, performance and timeliness are often crucial to a software system. The NFR-Framework [Chung93, Mylopoulos92] treats quality requirements 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. In this paper, we show how an historical record of the treatment of quality requirements during the development process can also serve to systematically support evolution of the software system. Changes can be driven by changes in organisational policy and changes in specific requirements. The effect of the changes can include a change in the target system and a change in the degree of meeting several requirements. We treat such changes in terms of (i) adding or modifying quality requirements, or changing their importance, and (ii) changes in design decisions and changes in design rationale. This incremental approach is illustrated by a study of changes in banking policies at Barclays Bank.

Classification:

Main Classification:

3.4: Reusing the artifacts of requirements engineering

C: Proposed process-oriented solution

Additional Classifications:

1.3: Coverting vague goals (e.g., "user-friendliness," "security") into specific properties

3.1: Ensuring that the results of requirements engineering are modifiable and maintainable.

2.1: Generating and evaluating alternative strategies for meeting requirements

Keywords: software quality, non-functional requirements, change, evolution, maintenance.

1 Introduction

Quality requirements (or *Non-Functional Requirements, NFRs*) such as confidentiality, performance and timeliness are often crucial to a software system, such as a banking information system. Furthermore, the continued success of a software system often requires meeting changing organisational needs in a competitive marketplace. In fact, changing and conflicting requirements were one of the three main problems for software development, as identified in problem-driven empirical studies [Curtis88]. Consequently, developers must continuously aim for higher software quality, increased productivity and lower cost, in a changing environment. But how can change be dealt with, given the need for quality (as embodied in the non-functional requirements)?

To systematically deal with change while aiming for high software quality, this paper adopts the NFR-Framework [Chung93, Mylopoulos92], and treats quality requirements as goals to be achieved during the

*The first author was at the Department of Computer Science, University of Toronto when a draft of this paper was prepared.

process of system development and change. Throughout the process, goals are decomposed, design trade-offs 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 a history record. In this paper, we show 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 operations), and changes in 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 — both the requirement which was changed, and others.

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.

This process-oriented incremental approach is illustrated by a study of dealing with quality in a changing bank-loan system. In the study, we consider the impact of changes in requirements, primarily those driven by changes in organisational policy.

Scenario. The scenario is based on two editions of *The Barclays Code of Business Banking* [Barclays92, 93a] which is in real-world use, setting out the general principles and terms of Barclays Bank's dealings with its business customers. One edition of the Code [Barclays92] was a common example for the Workshop on Research Issues in the Intersection between Software Engineering and Artificial Intelligence, Sorrento, Italy, May 1994.

In reviewing both editions of the Code [Barclays92,93a], 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 [Barclays92] 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 trend towards greater informativeness continued in the second edition [Barclays93a], which is quite similar to the first edition.

We consider how our approach might be used to help Barclays deal with quality in the presence of changes. Given a set of requirements for informativeness, performance, etc., our study considers the impact of a set of changes, primarily in organisational policy. Also reported are the resulting changes in achievement of goals and the impact on the system design.

Recall that the NFR-Framework supports the requirements engineering process in terms of goals, decisions, rationale and the like. These components are already included in the historical record, which facilitates our use of the framework for dealing with change. We treat both the sources of change (e.g., changes in organisational policy) and the effects of change (e.g., changes in the target system produced) in the historical record; there, they are treated in terms of changes in goals, changes in the importance of goals, changes in decisions, and changes in rationale. For example, adding an organisational requirement may result in adding a goal to the historical record.

This paper, then, shows how the NFR-Framework can be used to deal with change, illustrates the approach using the Barclays Bank example, and offers 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 generate an initial design from the bank's initial requirements. Section 3 shows how the initial design can be changed to deal with changes in requirements, based on the trends of change in the Barclays Codes. Section 4 describes guidelines for achieving desirable properties that an historical goal record should possess when dealing with changes. Section 5 discusses related work, and Section 6 summarises the contribution of the paper, lessons learned through the study, and some open issues.

Comparison with Previous Papers. This paper extends the usage of the NFR-Framework [Mylopoulos92, Chung93] by using its facilities to deal with *change*. We offer herein guidelines for well-

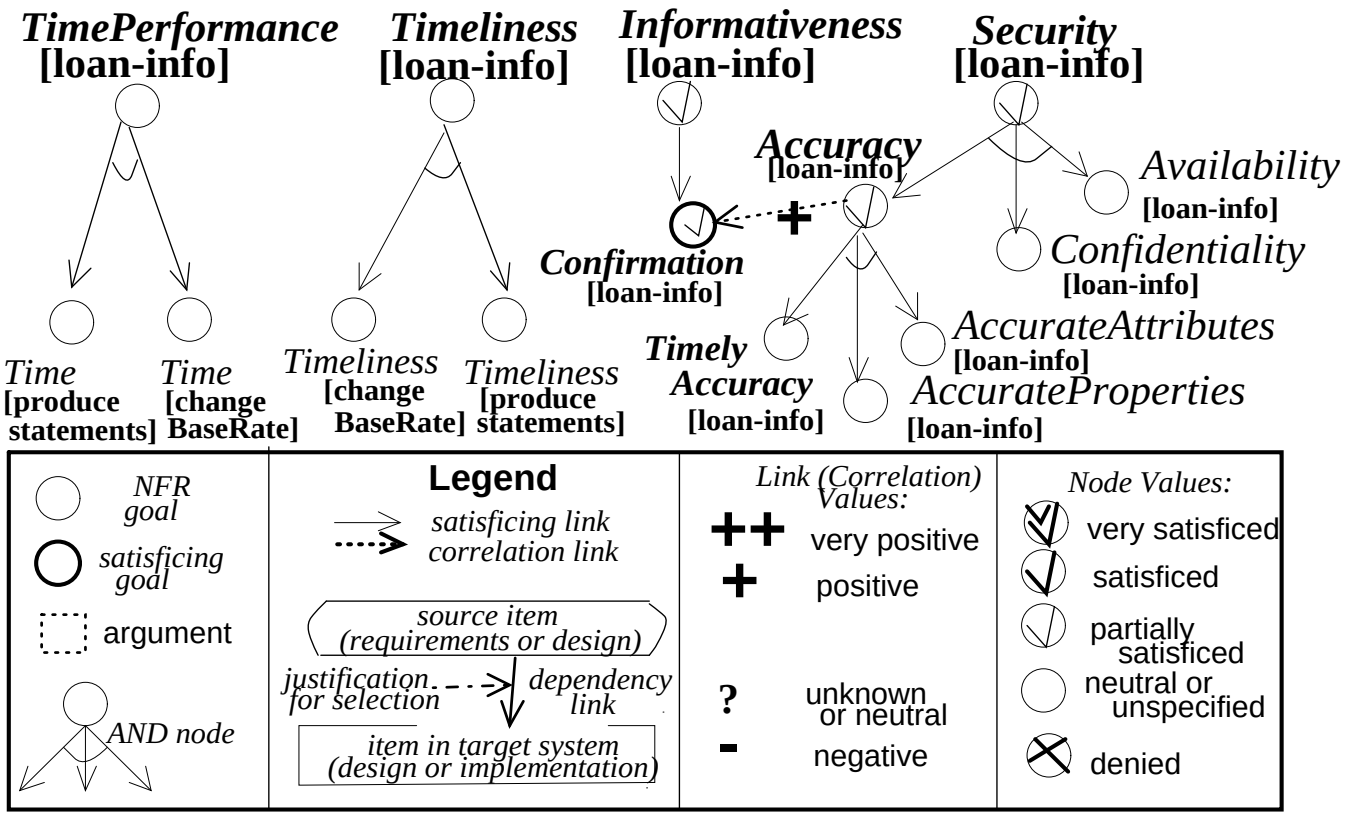

Figure 1: Initial top-level goal graph for loan information.

formedness of historical goal records in the presence of change. While drawing on our previous studies of attaining quality during the development of banking systems [Nixon93; Chung93a,b,94a,b], this paper presents an initial study of dealing with *change* in a bank loan system, to offer initial evidence for the usefulness of our approach to dealing with change. We also deal with the combination of performance and requirements for accuracy, timeliness and informativeness, in more detail than presented in [Chung94a,b].

2 Generating the Initial Design from Initial Requirements

This section describes how an initial design of the bank system might be generated from the initial quality requirements *before* changes were made. It also introduces the NFR-Framework and its main components. Section 3 continues the study, with consideration of changes in the bank's requirements.

2.1 Initial Top-Level Requirements

We consider a development process which reflects the situation prior to the 1992 changes in the Barclays Code. A number of quality requirements, including accuracy, performance, timeliness and security, are identified and dealt with.

One type of service that Barclays Bank provides is lending money. When clients want to borrow money, they go through negotiations with the bank concerning loan period, interest rates, method of payment, etc. The bank can change these conditions during the loan period, and wants to keep clients informed in a timely manner. The bank also wants to keep clients' financial affairs confidential.

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. Advance notice is not given for changes in the base rate, which take effect immediately for all clients, and are advertised in

the national press.

In reviewing Barclays' policies, we identified important *NFR goals,* some of which are captured and shown in Figure 1 (adapted from [Chung94a,b]). The goal *Timeliness*`[loan-info]` means quickly notifying clients about loan information. Similarly, *Informativeness*`[loan-info]` means loan information should be informative. *TimePerformance*`[loan-info]` means loan information should be processed by the system with good time performance, i.e., rapidly, and *Security*`[loan-info]` means that loan information should be maintained securely. These requirements are shown at the top of Figure 1 as *NFR goals.* 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.

Decompositions are one type of *method* for taking a goal and generating offspring goals. For example, the parent goal *Timeliness*`[loan-info]` can be decomposed into offspring goals *Timeliness*`[change BaseRate]` and *Timeliness*`[produce Statements]`, which are connected by an *AND* link. The interpretation is that if the base rate can be changed in a timely manner, *and* statements can be produced in a timely manner, *then* we have some confidence that timeliness of loan information can be "satisficed."[1] Besides *AND* links, other kinds of *link types* (beyond [Nilsson71]) are available, which indicate different relationships; for example a negative link (shown as "-" in figures) indicates that if the offspring goal is satisficed, the parent goal will be denied.

Goals can be decomposed using a variety of *methods.* Generic methods often capture knowledge from a domain or a quality-related field of study. Above, we used a method which decomposed *Timeliness*`[loan-info]` into goals for operations on the loan information. The same pre-defined generic method is used to decompose the top-level *TimePerformance* goal. In addition, developers can define and use their own methods. Using another method, the security goal is decomposed into goals for accuracy, confidentiality and availability, which are the major components of security as viewed by security experts [ITSEC91] [Clark87] [TCSEC85]. Similarly, accuracy can be decomposed [Chung93a] into goals for accurate attributes, accurate properties, and timely accuracy. Timely accuracy, the timely recording of information items in the information system, will be further considered in Section 3.

A *satisficing goal* is one which is intended to satisfice an NFR goal. It takes its parent goal and moves towards an alternative for the target system, and is shown as a dark circle. Barclays policies state that loan information be confirmed in writing for informativeness. The satisficing goal of *confirmation* is a good choice, not only for informativeness, but also for accuracy. Accordingly, a positive contribution of confirmation towards accuracy is established through a *correlation link* (shown as a dotted line) labelled with "+".

Each of the goals is associated with a *label* indicating the degree to which it is satisficed ("$\surd$") or denied ("**X**"). The *labelling algorithm* can be viewed as starting with leaf-nodes, and propagating values upwards, by considering the combination of node label values and link types. Here, satisficing confirmation gives some positive contribution to satisficing informativeness and accuracy, which in turn contributes towards satisficing security. We call the kind of graph shown in Figure 1 a "goal graph."

2.2 Generating the Initial Design

To generate an initial design (Figure 2), we continue the process of dealing with the initial quality requirements, focussing on a portion of Figure 1 which deals with timeliness and time performance. As mentioned above, changes in base interest rates[2] must be put into effect immediately. We consider a change in base rate prior to 1992 (After 1992, this discussion also applies, but as we will see in Section 3, it will interact with changes in other requirements).

The need to identify and focus on *critical* goals has been pointed out for performance goals in particular [Smith90], and quality goals in general [McCabe87] [Juran79]. Thus we refine the time performance goal for changing the base rate by identifying it as critical; to support this, we attach an argument *(Claim)* that base rate changes must be done immediately. Note that this design rationale in this case is based on organisational policy. Now for changing the base rate, the goal for *timeliness,* the provision of information in a timely way, will be helped by quickly changing the base rate. The reason is that a system which can make the change with good time performance will be able to assist in notifying customers of the change in

[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* [Simon1981] to suggest that generated software is expected to satisfy NFRs within acceptable limits, rather than absolutely.

[2] Note that a change in interest rates is not treated herein as a change in requirements.

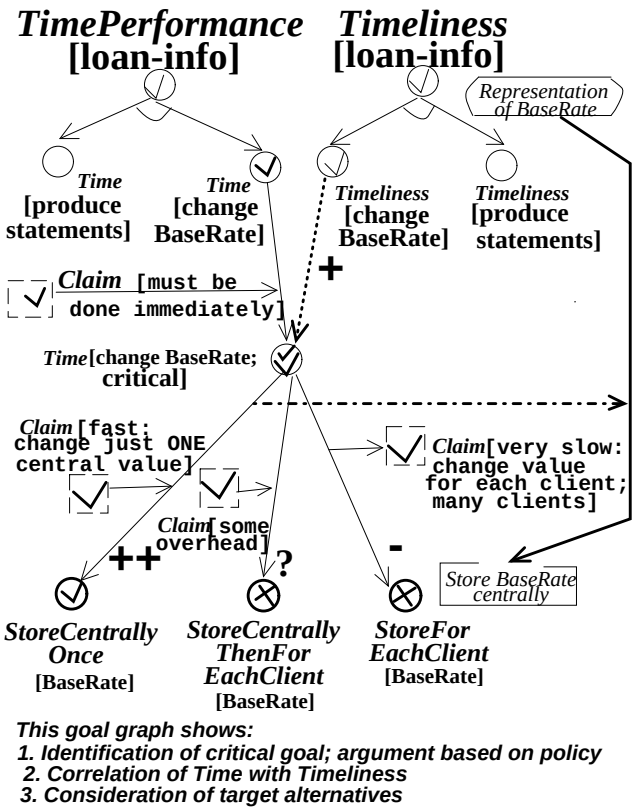

Figure 2: Goal graph with portion of initial design of loan information system.

a timely manner. This *correlation link* is detected and shown in the figure with a "+" sign.

In moving towards a target system for changing the base rate, we consider the *satisficing goals.* One alternative is to store the new rate for each client. This is ruled out by arguing that it is very slow (shown as "-" in Figure 2), since each client record must be updated, and that there are many clients. This later argument is based on *workload,* and is suggested by statistics [Barclays93b,c] available to us.[3] Although this alternative is negative for time performance, the change in requirements in Section 3 will introduce a tradeoff which shows a positive correlation with another quality requirement.

Another target alternative is to store the base rate in one central value in the information system. As only one value must be changed, it is argued that this provides a very fast method (shown as "++") of bringing the new base rate into effect, and this method is chosen for the target system. A third alternative is a hybrid of the other two methods; it involves first changing the centrally stored rate, then updating the rates stored in all client records. Due to overhead, it offers intermediate time performance (shown as "?") and has more overhead than only storing the value centrally; thus it is not selected now, but will be considered again in Section 3.

Now we have labelled some leaf nodes as satisficed or denied. 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 a good satisfaction of the critical time goal. This in turn has some positive contribution to the timeliness goal for changing the base rate, and some partial positive contribution to the overall goals for time performance and timeliness, shown at the top of Figure 2. Note that a satisficed offspring, when combined with a *negative* link, would lead to a denied parent; for example, if we had chosen *StoreForEachClient*, the critical time goal would have been denied.

Shown at the right of Figure 2 is a *dependency link* [Chung91b], which relates a source feature (here, the base rate) to a selected corresponding target-system feature (here, central rate storage), and also indicates a justification for the selection.

The process of constructing the goal graph was facilitated by the use of the framework's methods for capturing knowledge about decomposing and satisficing NFR goals as well as knowledge about justifying design decisions. This process was also facilitated by the use of the framework's correlation rules for capturing knowledge about goal interactions, positive and negative. The framework offers the evaluation procedure, by which we were also able to see, throughout this goal graph expansion process, the effect of each design decision using the framework's evaluation procedure.

We have now considered the use of the NFR-Framework, but without dealing with a change in requirements. The next section will consider the impact of changing requirements.

3 Changing the Design to Deal with Changed Requirements

We now consider how we could deal with a revision in requirements.

[3] 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.

3.1 Adding a New Requirement

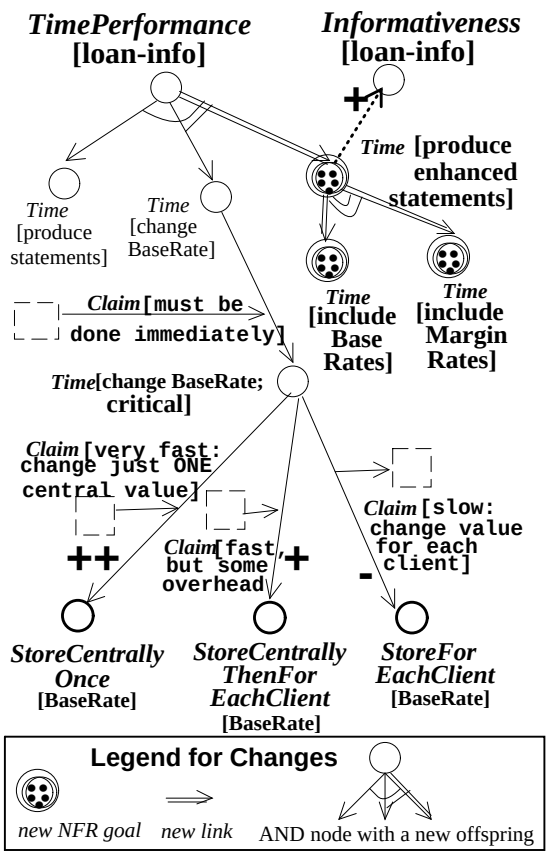

Figure 3: Goal graph with new requirement.

We consider a change in requirements, in particular the addition of a new requirement due to a change in organisational policy. We now consider the transition to offering loan clients more options for reporting on their statements. The provision of enhanced statements started in 1992. Both editions of the Code [Barclays92,93a] reflect the changes in organisational policy, with the trends being towards greater informativeness, and the provision of more detailed breakdown of charges.

For the purposes of our study, we assume that the provision of enhanced statement would include a detailed list of the base rate(s) and interest margin(s) which were in effect for the client during the period of the statement.

In Figure 3, we show the *addition* of a requirement for good time performance for producing enhanced statements. This 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 new links to offspring are presented.

3.2 Changing the Design to Meet the New Requirement

In Figure 4, the new requirements result in *interaction* with existing requirements, leading to new tradeoffs (e.g., accuracy-performance), 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 (which are assumed to now include the base rates). This leads to a tradeoff between timely accuracy and time performance.

Figure 4 shows a goal for Timely Accuracy 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, in which a time performance goal of producing enhanced statements was decomposed into time goals for including the various interest rates in the statements.

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" [Chung93a]. 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 anyhow; since each record would include the base rates, little additional work would then be needed to obtain them. Due to overhead, reasonable time performance, but a little slower (shown as "+"), 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" [Chung93a]). Here there is an accuracy issue which stems from including the base rate in statements. Suppose the base rate is changed just be-

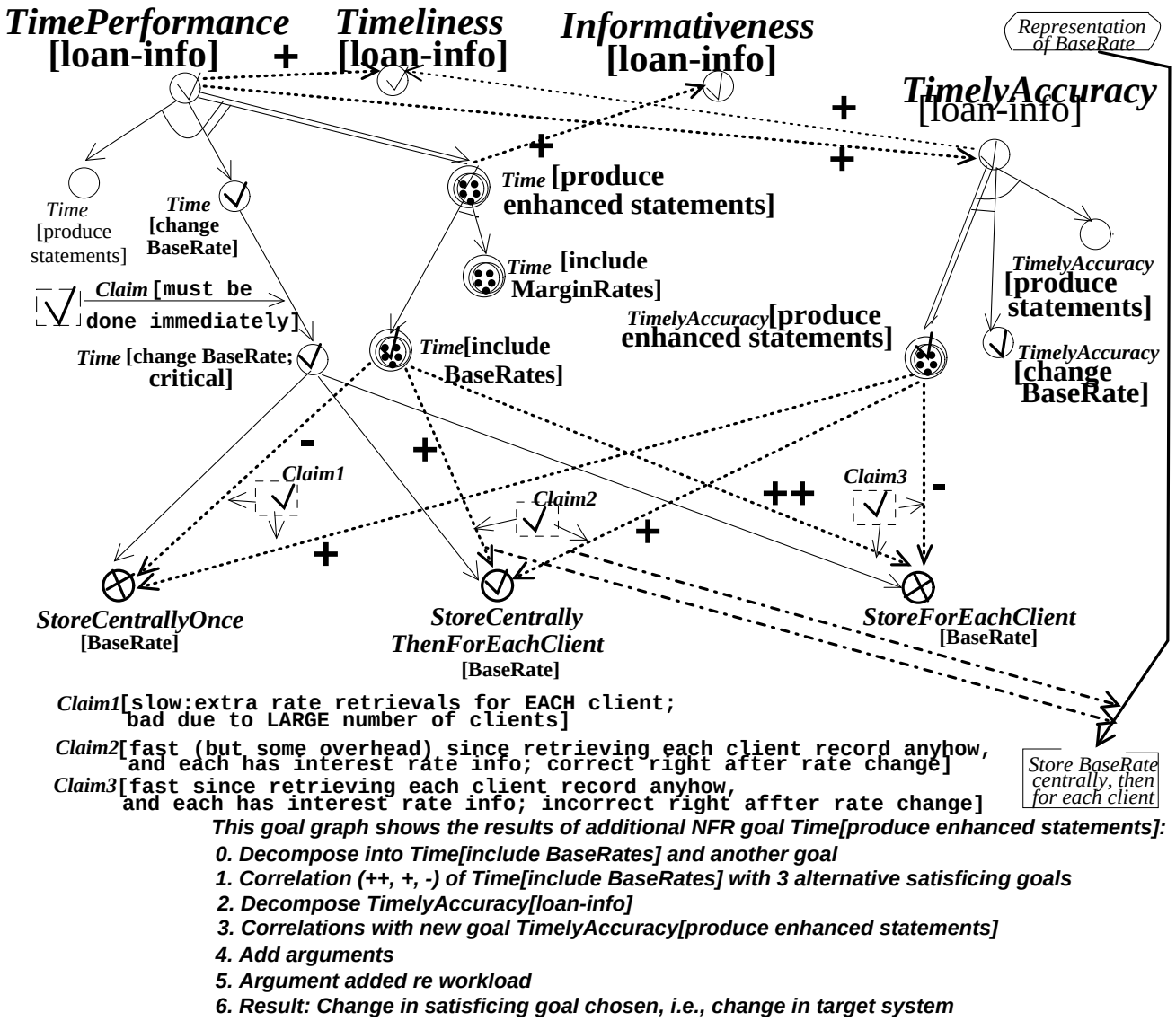

Figure 4: New design resulting from adding a new requirement.

fore 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 it is possible that some statements would 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 is ruled out, as it is generally slow, even though accurate. Thus the hybrid method is chosen, as it is accurate and offers reasonable performance. Recall that with the prior requirements (Figure 2), the hybrid method had been ruled out, due to performance overhead, and the rate would have been stored only centrally. Now, however, in the presence of the accuracy-performance tradeoff, the hybrid method is

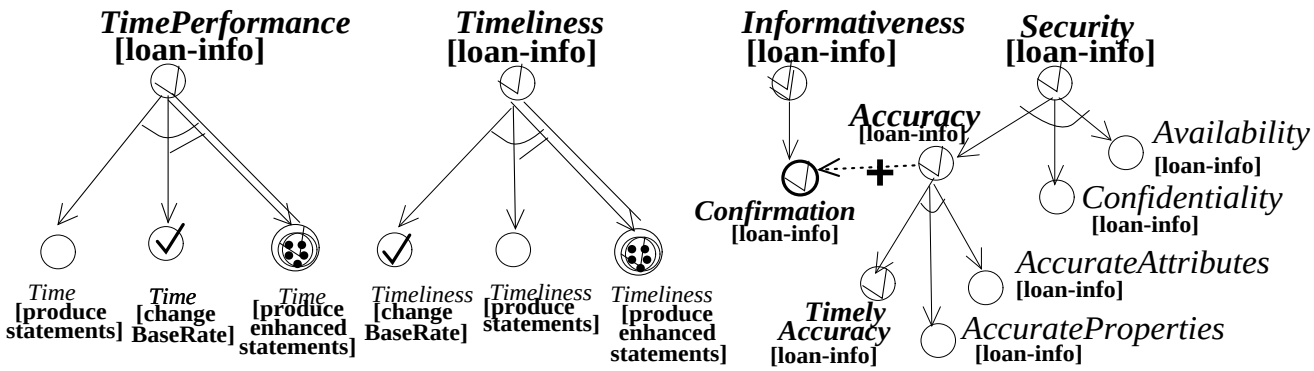

Figure 5: Impact on top-level requirements of adding new requirement.

still considered reasonable, offering timely accuracy, and also reasonable time performance for both changing the base rate and including the rate in statements. Thus consideration of an organisational change has resulted in a change in the target system, which is recorded as a revised dependency link.

Given the choice of target system, we wish to see the impact on quality requirements. We can use the framework's partially automated labelling procedure, which takes into account the values (satisficed, denied, etc.) of lower nodes, and the link relationships (positive or negative) to determine whether higher nodes are satisficed. 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 rate and including the base rates in statements. This last improvements has a positive impact on meeting ther time performance goal for producing enhanced statements. The overall impact on top goals will be shown in Figure 5.

3.3 Impact of Change on Other Top-Level Requirements

What has been the impact of the change in requirements?

1. *The goal graph (Figure 5) now includes the history of dealing with the change.* Goals have been added for producing enhanced statements, and offspring goals have been added for including various interest rates in the statements. In addition, the graph now contains the interactions which were detected 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,* not just the lower-level ones considered in the previous sub-section. The effect was to meet the new goals, which contributed *positively* towards the top-level goals. For example, the new goal *TimelyAccuracy*`[produce enhanced statements]` was satisficed and makes a positive contribution towards *TimelyAccuracy*`[loan-info]`. As part of the tradeoff, however, time performance for changing the base rate went from being very satisficed to being satisficed, which is still acceptable for this critical goal. By comparing Figure 5 with Figures 1 and 2, it can be seen that the overall impact has been an increase in informativeness and 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) are not decomposed to satisficing goals, or are not labelled. For example, we did not consider here how to satisfice the new time performance goal for including margin rates in statements. Likewise we did not consider, either initially or subsequently, *Time*`[produce statements]`. But for achieving good time performance for loan information, quickly including the margin rate in the enhanced statements *is* important, and quickly producing the basic statements *was* and *is* important. So here, without satisficing the offspring we cannot label the top-level parent *TimePerformance*`[loan-info]` as satisficed. What we can say about the 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 Treatment of Revised Requirements

Incorporating changes into the initial design involves identifying changes in the requirements, organisation and its workload, and associating them with appropriate components of the history record, i.e., goal graph. This process of identification and association is often interleaved, and proceeds incrementally, together with the analysis of change in design tradeoffs, the evaluation of change in goal achievement, and the rationalisation of the change in relation to the previous system and a trace of its sources.

An important aspect of our work is its knowledge-based approach, helping the developer capture, use and re-use development knowledge. To more fully realise the benefits of this approach, tools should provide semi-automated support of a developer dealing with change. We have developed tools for dealing with quality requirements for the NFR-Framework.[4] These tools primarily help the developer state and refine goals. But now, dealing with change may require operations, such as removal of a goal, which may temporarily make the goal graph inconsistent. In these cases, help is needed to restore the goal graph to a consistent state. Accordingly we offer some initial guidelines, for helping ensure that some syntactic and semantic structural consistency criteria are met. These would help us provide a tool to deal with situations relating to change, some of which were illustrated in Section 3.

4.1 Desirable Properties of Goal Graphs

Incorporation of revisions into a goal graph includes a process of refining goals, detecting interactions, disambiguating terminology, and detecting conflicts, omissions and redundancies. This process 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:

1. Each argument should be attached to a goal or a link.

2. 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.

This knowledge-based process becomes more justifiable by the use of the notion of a *knowledge-source traceable goal graph:*

1. Each decomposition, satisficing or argumentation should be associated with either a generic- or developer-supplied method.

2. Each correlation link should be associated with a generic- or user-supplied correlation rule.

These guidelines help trace decisions to identifiable sources of knowledge, including generic knowledge of a particular NFR (e.g., security or performance), and the developer's expertise.

When a goal graph is completed with respect to the incoporation of changes, we use the notion of a *goal-decision traceable goal graph:*

1. Each NFR goal should have at least one target-system design alternative selected for it; conversely, a design alternative needs to be traceable to a NFR goal.

2. Each goal, at least important ones, needs to be associated with an originator, such as a responsible person or a document.

This notion allows us to trace the relationships among design decisions, their corresponding goals, and the people who stated the goals.

The process of change is incremental, i.e., there is a goal graph before the change, and changes are reflected into the existing goal graph, which results in a new goal graph. However, the construction of a meaningfully revised goal graph may proceed at intervals after a group of changes are introduced in order to allow consideration of a temporarily partially disconnected goal graph. We thus introduce the notion of a *meaningfully-changed goal graph:*

1. When a change process is completed, each sink (i.e., a node without any outgoing link) should to be labelled as either satisficed or denied, and the labelling procedure invoked.

2. An argument or an originator should be recorded for a change.[5]

[4] The NFR-Assistant [Chung93a,b] has been implemented to deal with security, accuracy and other NFRs. Work is in progress on the Performance Assistant [Nixon94].

[5] This is suggested by the second criterion for goal-decision traceable goal graphs.

3. In addition, although omitted from goals graph presentations, the times of changes should be recorded[6].

This notion helps ensure that the reasons for changes are documented, and that the results of the changes are evaluated and can be compared to the previous system.

4.2 Guidelines for Different Categories of Changes

Incorporating revisions into a goal graph is facilitated by a scheme which divides NFR-related changes into three categories (which correspond 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:

(a) *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.

(b) *deletion of a quality requirement:* This can involve deleting (possibly recursively) all links (both incoming and outgoing) associated with the requirement, or re-associating the links with another goal. Each satisficing goal without a parent should be deleted or linked to another NFR goal. An argument without a parent usually needs to be deleted.

(c) *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:

(a) *addition:* This establishes new correlation links and affects goal achievement.

(b) *deletion:* If this results in an NFR goal having no remaining satisficing goal, that NFR goal needs further expansion. Also, some correlation links may be deleted, which affects goal achievement.

(c) *change in criticality:* This affects goal achievement.

Furthermore, if a different satisficing goal is selected for (or dropped from) the target system, dependency links will have to be revised.

Guidelines for Changes in Arguments.

(a) *addition:* If the argument supports another goal, it has the effect of strengthening the importance of the goal; if the argument denies another goal, it would have the opposite effect.

(b) *deletion:* If the argument is in support of another goal, it has the effect of weakening the importance of the goal; otherwise, it would have the opposite effect.

(c) *change in criticality:* (1.) If the argument is in support of another goal, this has the effect of strengthening the importance of the goal, if the argument becomes more critical, but weakens its importance if the argument becomes less critical. (2.) If the argument is in denial of another goal, this has the effect of weakening importance of the goal, if the argument becomes more critical, but strengthens its importance if the argument becomes less critical.

5 Literature Review

Related work includes work in decision support systems (e.g., [Lee91] [Potts88]) 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

[6] We can represent a goal graph in a conceptual modelling language such as Telos [Mylopoulos91] which allows the association of various times of changes, such as creation time and deletion time, by treating links as objects which have attributes.

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 [Gotel94] 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, such as ambiguities, omissions and conflicts. However, we have looked at traceability with more emphasis on incorporating changes in NFRs and 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 [Basili88] 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 TAME's goals are general requirements and uses a collection of metrics for improvement, while our approach uses 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 [Coleman94] can also play an important role in maintenance, and our work is generally related to work in maintenance on software evolution and traceability.

Work on requirements analysis for safety-critical systems [Coombes94] shares a similar spirit with ours in that both are concerned with NFRs and their analysis starting with organization needs. Coombes and McDermid's work has taken a causal reasoning approach, hence is complementary to our amalgamation of qualitative reasoning from truth maintenance systems [AI84] and dialectical reasoning from work on design rationale [Lee91]. The two are also complementary as Coombes and McDermid's work focusses on safety and its analysis, while our approach is directed to systematically supporting changes using NFRs.

Requirements engineering research can benefit from the study of real-world systems [Fickas94]. We have benefitted from this approach in reviewing editions of *The Barclays Code of Business Banking.*

This work grows out of the earlier DAIDA environment [Jarke92a] 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 tools using DAIDA's knowledge base management facilities [Jarke92b]. Our framework follows a decision-oriented (as opposed to activity- or product-oriented) approach to managing software evolution [Jarke88] [Rolland94], but provides additional representation and support, and is driven by quality requirements.

6 Conclusions

Our Observations in this Study. Let us now give our observations in this study. In dealing with change, our operations on the goal graphs (going from Figure 2 to subsequent figures) has mainly involved the following operations (from Section 4):

- adding a new goal (e.g., due to changes in organisational policies);
- revising an existing interaction (e.g., correlations; labelling);
- considering new interactions;
- (re-) considering alternative target systems;
- 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* bookkeeping (e.g., some correlation and labelling) 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. [Greenspan82, 94]), which is needed, used, and re-used.

Future Work. Interestingly, this study has presented some support for some extensions to the NFR-Framework, particularly in representing changes in strengths of goal achievement. Examples have arisen for both increases and decreases in strength. For example, the organisational policy change resulted in an increase in informativeness, as well as a decrease in performance, albeit to acceptable levels. While the framework currently has a variety of values for goal achievement (e.g., satisficed, denied, neutral, etc.), this study suggests adding additional "shadings" to the *existing* values. For example, there could be several degrees of satisficed, and several degrees of denied. This 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 notiation would be helpful to the tools for display purposes.

Considering a vast amount of informal descriptions, sometimes loosely connected to one another, we feel that this study suggests the need for even more structure to argumentation, perhaps by way of a model of organisation which provides a rich ontology, as presented, for example, in [Yu93, 94a]

Another avenue of future work concerns some extensions to the NFR-Assistant, a tool intended to help developers use the NFR-Framework, to distinguish new goals and links from old ones in a goal graph, in extracting the differences between the old and new goal graphs, and in using the guidelines for incorporating changes. While we have presented in Section 4 some guidelines for dealing with individual operations on goal graphs, it would also be helpful to determine some commonly-arising sequences of operations, and then offer such sequences as atomic operations in a tool.

Further studies of real systems could be made, in order 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.

Conclusion. In dealing with change, our overall long-term goals are to increase quality, decrease developers' effort (by decreasing development time and cost), and to improve conciseness of representation and schemes for assisting reasoning. We have taken an existing framework [Chung93, Mylopoulos92] 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 in a quality-oriented manner.

More specifically, our approach leads to the following benefits:

- changes in NFRs are representable, thanks to knowledge-based approach which enables capturing concepts central to dealing with NFRs, both initial and subsequent ones;
- incorporating change is conceptually simple, thanks to the use of the same structure for developing and recording both the initial goal graph and its subsequent revisions;
- incremental change is dealt with quickly, thanks to the NFR-Framework's provisions for encoding NFR-specific information to relate new requirements to old ones, and to find correlations among the new requirements.
- identifying the needed change is less time-consuming and less costly, thanks to the already encoded goals and their interactions. By consulting the initial goal graph, a developer dealing with change can detect patterns in goal graphs, and reduce searches of large volumes of documentation whose references to quality have already been captured in the initial goal graph.

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. Most work on change and evolution deals with the historical or temporal dimension, e.g., versioning and configuration management. Future work would explore the interaction of the means-ends reasoning dimension with these other dimensions in dealing with change. The idea of applying means-ends reasoning to deal with change is also the basis for a framework for process reengineering [Yu94a,b, forthcoming], where a combination of functional and non-functional means-ends reasoning is used to support process re-design.

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

Bibliography

[AI84] *Artificial Intelligence Journal,* Special Issue on Qualitative Reasoning, vol. 24, nos. 1–3, Dec., 1984.

[Barclays92] Barclays Bank PLC, *The Barclays Code of Business Banking.* London, England, effective 31st January 1992.

[Barclays93a] Barclays Bank PLC, *The Barclays Code of Business Banking.* London, England, May 1993.

[Barclays93b] Barclays Bank PLC, *Annual Review & Summary Financial Statement,* London, England, 1993.

[Barclays93c] Barclays Bank PLC, *Report and Accounts,* London, England, 1993.

[Basili88] V. R. Basili and H. D. Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments," *IEEE Transactions on Software Engineering,* vol. 14, no. 6, June 1988, pp. 758–773

[Chung91a] Lawrence Chung, "Representation and Utilization of Non-Functional Requirements for Information System Design." In R. Anderson, J. A. Bubenko, Jr., A. Sølvberg (Editors), *Advanced Information Systems Engineering,* Proc., 3rd Int. Conf. CAiSE '91, Trondheim, Norway, May 13–15, 1991. Berlin: Springer-Verlag, 1991, pp. 5–30.

[Chung91b] K. Lawrence Chung, Panagiotis Katalagarianos, Manolis Marakakis, Michalis Mertikas, John Mylopoulos and Yannis Vassiliou, "From Information System Requirements to Designs: A Mapping Framework." *Information Systems,* Vol. 16, No. 4, 1991, pp. 429–461.

[Chung93a] Kyungwha Lawrence 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.

[Chung93b] Lawrence Chung, "Dealing With Security Requirements During the Development of Information Systems." In Colette Rolland, François Bodat and Corine Cauvet (Editors), *Advanced Information Systems Engineering,* Proc., 5th Int. Conf. CAiSE '93, Paris, France, June 8–11, 1993. Berlin: Springer-Verlag, 1993, pp. 234–251.

[Chung94a] Lawrence Chung, Brian A. Nixon and Eric Yu, Using Quality Requirements to Drive Software Development. Presented at the *Workshop on Research Issues in the Intersection Between Software Engineering and Artificial Intelligence,* Sorrento, Italy, May 16–17, 1994.

[Chung94b] Lawrence Chung, Brian A. Nixon and Eric Yu, Using Quality Requirements to Systematically Develop Quality Software. To appear in *Proceedings, Fourth International Conference on Software Quality,* McLean, VA, U.S.A. October 3–5, 1994.

[Chung94c] Lawrence Chung and Brian A. Nixon, Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach. Working paper, July 1994.

[Clark87] D. D. Clark and D. R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," *Proc. IEEE Symposium on Security and Privacy,* 1987, pp. 184–194.

[Coleman94] Don Coleman, Dan Ash, Bruce Lowther and Paul Oman, "Using Metrics to Evaluate Software System Maintainability." *IEEE Computer,* vol. 27, no. 8, August 1994, pp. 44–49.

[Coombes94] A. C. Coombes and J. A, McDermid, "Using Qualitative Physics in Requirements Specification of Safety Critical Systems — A Potential AI User's Perspective." Presented at the *Workshop on Research Issues in the Intersection Between Software Engineering and Artificial Intelligence,* Sorrento, Italy, May 16–17, 1994.

[Curtis88] Bill Curtis, Herb Krasner and Neil Iscoe, *A Field Study of the Software Design Process for Large Systems. Communications of the ACM,* vol. 31, no. 11, November 1988, pp. 1268–1287.

[Fickas94] Stephen Fickas, "Desert Island Column," *Automated Software Engineering,* Vol. 1, No. 1, March 1994, pp. 141–146.

[Finkelstein93] Anthony C. W. Finkelstein and Stewart J. M. Green, *Goal-oriented Requirements Engineering.* Technical Report TR–93–42, Imperial College (London Univ.), forthcoming.

[Gotel94] Orlena C. Z. Gotel and Anthony C. W. Finkelstein, An Analysis of the Requirements Traceability Problem. *Proceedings, International Conference on Requirements Engineering,* Colorado Springs, 1994. IEEE Computer Society Press, 1994.

[Greenspan82] S. J. Greenspan, J. Mylopoulos and A. Borgida, "Capturing More World Knowledge in the Requirements Specification." *Proceedings, Sixth International Conference on Software Engineering,* 1982, pp. 225–234.

[Greenspan94] Sol Greenspan, John Mylopoulos and Alex Borgida, "On Formal Requirements Modeling Languages: RML Revisited." *Proceedings, 16th International Conference on Software Engineering,* Sorrento, Italy, May 1994, pp. 135–147.

[ITSEC91] Office for Official Publications of the European Communities, *Information Technology Security Evaluation Criteria, Provisional Harmonised Criteria,* Version 1.2, June 1991, Luxembourg.

[Jarke88] Matthias Jarke and Thomas Rose, "Managing Knowledge about Information System Evolution." *Proceedings, SIGMOD '88,* Chicago, May 1988, pp. 303–311.

[Jarke92a] M. Jarke, J. Mylopoulos, J. W. Schmidt, Y. Vassiliou, "DAIDA: An Environment for Evolving Information Systems," *ACM Trans. Information Systems*, vol. 10, no. 1, Jan. 1992, pp. 1–50.

[Jarke92b] Matthias Jarke (Editor), *ConceptBase V3.1 User Manual.* Univ. of Passau, 1992.

[Juran79] J. M. Juran, Frank M. Gryna Jr., and R. S. Bingham Jr. (Eds.), *Quality Control Handbook,*, 3rd Ed., New York: McGraw-Hill Book, 1979.

[Lee91] Jintae Lee, Extending the Potts and Bruns Model for Recording Design Rationale. *Proc., 13th Int. Conf. on Software Eng.,* Austin, Texas, May 13–17, 1991, pp. 114–125.

[McCabe87] Thomas J. McCabe and G. Gordon Schulmeyer, "The Pareto Principle Applied to Software Quality Assurance," In G. Gordon Schulmeyer and James I. McManus (Eds.) *Handbook of Software Quality Assurance,* New York: Van Nostrand Reinhold, 1987, pp. 178–210.

[Mylopoulos91] John Mylopoulos, Alex Borgida, Matthias Jarke, and Manolis Koubarakis, Telos: Representing Knowledge about Information Systems, *ACM Transactions on Information Systems*, vol. 8, Oct. 1990, pp. 325–362.

[Mylopoulos92] John Mylopoulos, Lawrence Chung and Brian Nixon, "Representing and Using Non-Functional Requirements: A Process-Oriented Approach." *IEEE Trans. on Software Eng.,* Special Issue on Knowledge Representation and Reasoning in Software Development, Vol. 18, No. 6, June 1992, pp. 483–497.

[Mylopoulos93] John Mylopoulos, Lawrence Chung, Eric Yu and Brian Nixon, *Requirements Engineering 1993: Selected Papers.* Technical Report DKBS–TR–93–2, Dept. of Computer Science, Univ. of Toronto, July 1993.

[Nilsson71] Nils Nilsson, *Problem-Solving Methods in Artificial Intelligence.* New York, McGraw-Hill, 1971.

[Nixon91] Brian Nixon, "Implementation of Information System Design Specifications: A Performance Perspective." In Paris Kanellakis and Joachim W. Schmidt (Eds.), *Database Programming Languages: Bulk Types & Persistent Data — The 3rd International Workshop.* Aug. 27–30, 1991, Nafplion, Greece. San Mateo, CA: Morgan Kaufmann, pp. 149–168.

[Nixon93] Brian A. Nixon, "Dealing with Performance Requirements During the Development of Information Systems." *Proc. of the IEEE Int. Symp. on Requirements Eng.,* San Diego, CA, January 4–6, 1993. Los Alamitos, CA: IEEE Computer Society Press, pp. 42–49.

[Nixon94] Brian A. Nixon, "Representing and Using Performance Requirements During the Development of Information Systems." In Matthias Jarke, Janis Bubenko, Keith Jeffery (Eds.), *Advances in Database Technology - EDBT '94,* 4th International Conference on Extending Database Technology, Cambridge, United Kingdom, March 1994, Proceedings. Berlin: Springer-Verlag, 1994, pp. 187-200.

[Potts88] Colin Potts and Glenn Bruns, Recording the Reasons for Design Decisions, *Proc., 10th International Conference on Software Engineering,* 1988, pp. 418–427.

[Rolland94] Colette Rolland "Modeling the evolution of artifacts." *Proceedings, International Conference on Requirements Engineering,* Colorado Springs, Colorado, U.S.A., April 1994.

[Simon81] Herbert A. Simon, *The Sciences of the Artificial,* Second Edition. Cambridge, MA: The MIT Press, 1981.

[Smith90] Connie U. Smith, *Performance Engineering of Software Systems.* Reading, MA: Addison-Wesley, 1990.

[TCSEC85] U.S. Department of Defense, *Trusted Computer Systems Evaluation Criteria,* DOD 5200.28–STD, Dec. 1985.

[Yu93] Eric S. K. Yu, Modelling Organizations for Information Systems Requirements Engineering. *Proc. of the IEEE Int. Symp. on Requirements Eng.,* San Diego, CA, January 4–6, 1993. Los Alamitos, CA: IEEE Computer Society Press, pp. 34–41.

[Yu94a] Eric S.K. Yu and John Mylopoulos, 'Understanding "Why" in Software Process Modelling, Analysis, and Design.' *Proceedings, 16th International Conference on Software Engineering,* Sorrento, Italy, May 1994, pp. 159–168. A longer version is available in Report DKBS–94–3, Dept. of Computer Science, University of Toronto, 1994.

[Yu94b] Eric S.K. Yu and John Mylopoulos, 'From E-R to "A-R" – Modelling Strategic Actor Relationships for Business Process Reengineering.' Accepted for *Proceedings, 13th International Conference on the Entity-Relationship Approach,* Manchester, U.K., December 1994.

[YuForthcoming] Eric Yu, "A Framework for Process Modelling and Reengineering." Ph.D. Thesis, Dept. of Computer Science, Univ. of Toronto, forthcoming.