*Research paper submitted to ICSE 1995.*
*September 1994.*

# Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach

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

Department of Computer Science
University of Toronto

Toronto, Ontario, Canada M5S 1A4
nixon@ai.toronto.edu
(416) 978–4299
Facsimile (416) 978–1455

## Abstract

*Quality characteristics are vital for the success of software systems. To remedy the problems inherent in ad hoc development, a framework has been developed to deal with non-functional requirements (quality requirements or NFRs). This framework takes the premise that the quality of a product depends on the quality of the process that leads from high-level NFRs to the product. The purpose of this paper is to give an initial evaluation of a recently proposed framework for dealing with non-functional requirements, by way of three studies of small portions of three information systems. These systems have a variety of characteristics, reflecting a variety of real application domains. This paper attempts to evaluate the extent to which the framework's objectives are met, namely, representing NFR-specific requirements, considering design tradeoffs, relating design decisions to NFRs, justifying the decisions, and assisting defect detection. The studies deal with and analyse three important classes of non-functional requirements for systems, namely, accuracy, security, and performance. The studies provide preliminary support for the usefulness of certain aspects of the framework, while raising some open issues. Some of the results have been reviewed by people with a knowledge of the kinds of organisations whose systems were studied.*

**Keywords:** non-functional requirements, software quality, defect detection.

*Initial versions of this paper were prepared when the first author was at the Department of Computer Science, University of Toronto.

## 1 Introduction

Quality characteristics, such as accuracy, security and performance, are vital for the success of software systems, such as credit card, medical record and taxation systems. As such systems become more and more central in our everyday life, so does our concern for their *quality*. For a credit card system, for example, it would be desirable to have account information maintained accurately, stored securely, and processed quickly, in a user-friendly manner. Quite appropriately, these concerns — *non-functional requirements* (quality requirements, hereafter *NFRs)* — have been recognized as playing a crucial role in requirements engineering [Jarke93] [Dardenne91] [Fickas87]. But how do we deal with such NFRs?

In the past, relatively little attention has been paid to the process of systematically dealing with NFRs, and developers have relied mostly on their own intuition, in an *ad hoc* way. For example, NFRs (e.g., how to securely [Benzel89] update an account) often are retrofitted late in the development process, or pursued in parallel with, but separately from, *functional* design (e.g., how to update an account). An *ad hoc* development process often makes it difficult to generate systems which meet user expectations, to justify the resulting systems, and to detect *defects* (or *faults,* such as conflicts and ambiguities in requirements [Feather93] and corresponding designs) early, although correction of design or implementation errors is much more costly than correction at the requirements phase [Boehm87].

To remedy the problems inherent in ad hoc development, a framework (the *NFR-Framework* [Mylopoulos92] [Chung93a]) has been developed to systematically deal with NFRs. This framework takes the

| | 1: Credit Card | 2: Health Insurance | 3: Gov't. Administration — Cabinet & Taxation — |
|---|---|---|---|
| **Domain Characteristics** | commercial<br><br>highly-competitive<br>very large transaction vol.<br>internal & external op. | commercial, governmental, professional<br>continuous improvement<br>large transaction vol.<br>emphasis on external op. | governmental<br>longer-term, consultative<br>relatively stable<br>lower transaction vol.<br>emphasis on internal op. |
| **NFRs (Sorts)** | *TimelyAccuracy*<br>*Performance*<br>*Security/Confidentiality* | *Confidentiality*<br>*(Accuracy)*<br>*(User-friendliness)* | *Confidentiality*<br>*Performance*<br>*(User-friendliness, Accuracy)* |

Table 1: Some characteristics of the three studies.

premise that the quality of a product depends largely on the quality of the process that leads from high-level NFRs to the product. The purpose of this paper is to give an initial evaluation of this process-oriented framework for dealing with NFRs. More specifically, this paper attempts to evaluate the extent to which the framework's objectives are met, namely, representing NFR-specific requirements, considering design tradeoffs, relating design decisions to NFRs, justifying the decisions, and assisting defect detection.

For this evaluation, we have carried out studies of small portions of three information systems, representing a variety of domains: credit cards, public health insurance, and government administration information systems (Table 1). These systems, ranging from commercial to governmental, showed considerable variation in domain characteristics, reflecting a variety of real application domains. This variation affected the types of NFRs stated, the types of design rationale used, and the number and cost of techniques used to meet the requirements. The studies dealt with three important classes of NFRs, namely *accuracy,* the faithful representation of information; *security,* the protection of information and system resources; and *performance,* the efficient and responsive use of system resources. The use of systems which do not meet these NFRs may lead to loss of money and trust, incorrect diagnosis and treatment, and inefficient administration.

The studies (Appendices A, B and C) are based on domain descriptions of existing systems, dealt with only *small portions* of systems, and were carried out "off-line," i.e., by examining documentation, without our involvement as actual participants during the development process. We ourselves did not develop the systems, but did deal with quality considerations for such systems. However, we tried to make minimal assumptions, and be faithful to the *source documents*, which varied in the amount of detail provided, and involved, for the government systems, consideration of proposed extensions to systems. Therefore, the primary purpose of the studies is to provide an *analysis of quality requirements* and the *development decisions* that might be made, given such requirements.

The twin starting points for this paper are the NFR-Framework and its tool (applications of which are reviewed in Section 2) and studies of credit card [Nixon93] [Chung93b], public health insurance [Chung93a], and government administration (Cabinet Documents [Chung93a] and Taxation Appeals [Nixon94]) information systems (Appendices A, B and C).

The present paper, then, extends this material with a methodology for using the NFR-Framework to deal with quality and detect defects (Section 3), and presents evaluations of the studies from the viewpoints of ourselves as framework developers and users (Section 4) and of domain experts[1] (Section 5). The final sections describe related work and concluding remarks.

## 2 Review and Illustration of NFR-Framework and Tool

The *NFR-Framework* [Mylopoulos92] [Chung93a] takes, in the spirit of rational design [Simon81], a *process-oriented approach* to dealing with NFRs, hence being complementary to the traditional product-oriented approach whose emphasis lies in product evaluation, usually involving metrics. The framework was influenced by work on decision support systems [Lee91] [Hahn91] [Potts88] and the DAIDA environment for information system engineering [Jarke92a].

[1]To obtain feedback, both positive and negative, we have circulated an earlier draft of this paper along with a questionnaire we have developed on quality requirements. We report herein on interviews with experts familiar with two of the organisations studied. We have also made initial contacts for future interviews with others.

*During* the software development process, this framework allows treating NFRs as potentially conflicting or synergistic goals to achieve, while considering development alternatives which could meet the stated NFRs, examining design tradeoffs, relating design decisions to NFRs, justifying the decisions in relation to the needs of the intended application domain, and assisting defect detection. In the framework, NFRs are an integral part of the entire software development process, used to drive the process, and act as selection criteria. A prototype tool, the *NFR-Assistant,* is also available and offers several functions and corresponding interfaces to assist the developer in this process.

To illustrate how a developer might use the NFR-Framework and the NFR-Assistant to develop a quality system, including our studies, the example of this section draws on the domain description of one of the Government Administration systems, which are dealt with in Appendix C.

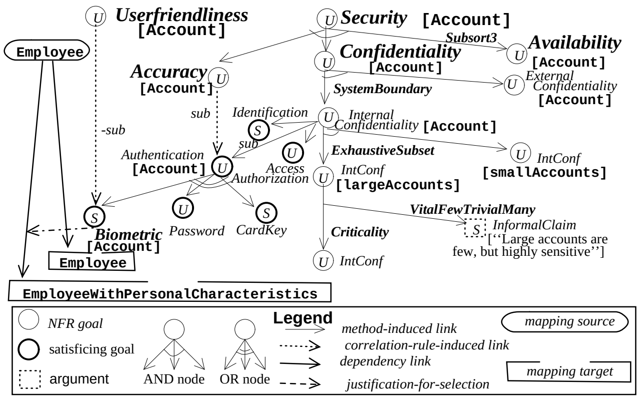


Figure 1: Goal graph structure for secure document.

For the purposes of illustration, here it suffices to say that there is a security requirement, "all documents should be secure." The developer represents this NFR as a goal *Security*[`Document`], shown at the top of a *goal graph structure* (See Figure 1). This is similar in spirit to AND/OR trees [Nilsson71], but has the notion of goal *satisficing* [Simon81][2] which assumes that development decisions usually contribute only partially towards (or against) a particular goal, rarely "accomplishing" or "satisfying" goals in a clear-cut sense. Consequently generated software is expected to satisfy NFRs within acceptable limits, rather than absolutely. In the framework, a goal graph structure is used to record and structure NFRs, design alternatives, decisions and rationale. All of these are uniformly treated as goals (denoted by nodes) and related to one another via links.

In this illustration, the developer starts with an initial set of security and other types of NFR goals, and then refines them into other NFR goals. To satisfice NFR goals, the developer considers design alternatives, called *satisficing goals*, along with their tradeoffs, refines them, makes selections and justifies them by recording design rationale, called *argumentation goals*. Throughout this process, the impact of each design decision is propagated towards top-level NFR goals, via *labels* indicating the degree of satisfaction.

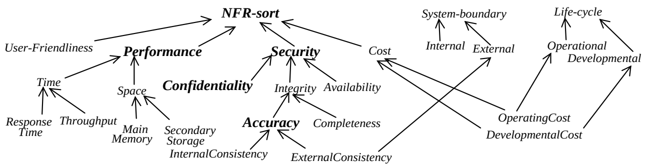


Figure 2: Sort hierarchy.

Through its *components*, the framework aims to make the development process less *ad hoc* and advocates a more formal, systematic methodology:

NFRs are often associated with a rich, but potentially confusing set of concepts (*sorts*), which are organised into an hierarchy (Figure 2) to guide the developer.

Development knowledge about specific NFRs are to be taken from the literature and industrial experience and captured as *methods,* which are then presented for reuse to help the developer generate new goals and links. For example, techniques can be incorporated from security evaluation criteria [ITSEC91], performance responsiveness principles [Smith90], and accuracy concepts [Motro89] [Martin73].

Non-functional requirements are systematically integrated into the development. They are represented as potentially conflicting or synergistic *goals,* which are intended to serve as a class of criteria for selecting among development alternatives for functional requirements, and justify the overall development, thereby becoming an integral part of the development process.

In the top goal of Figure 1, for instance, *Security* is the sort and `Document` is the *parameter* which typically represents information items such as attributes [Chung91a,b]. Now the developer can decompose the goal either on its sort or on its parameter.[3] We used

[2]Simon actually uses the term to refer to decision methods that look for satisfactory solutions rather than optimal ones. Similarly, the notion of *assurance* [ELee92] refers to satisfying security requirements within acceptable limits.

[3]In figures, some sort names are abbreviated and not all link types are shown.

the *Subsort* method to decompose the goal into three other goals for *integrity* (guarding against unauthorized update or tampering), *confidentiality* (guarding against unauthorized disclosure), and *availability* (guarding against interruption of service) of the document. The developer successively generates more specific goals (the *offspring*), each of which is generated with the aim of satisficing the *parent* goal.

Figure 3 shows the main control window for the NFR-Assistant (taken from [Chung93a] to show its use for developing a research expense management system), whose implementation uses the ConceptBase [Jarke92b] knowledge base management system. A developer uses the main control window to select a goal to refine and also to indicate the method to apply. Developers can either select built-in methods from a displayed hierarchy[4] to automatically generate offspring goals and parent-offspring links or, if no appropriate one is found, define their own methods

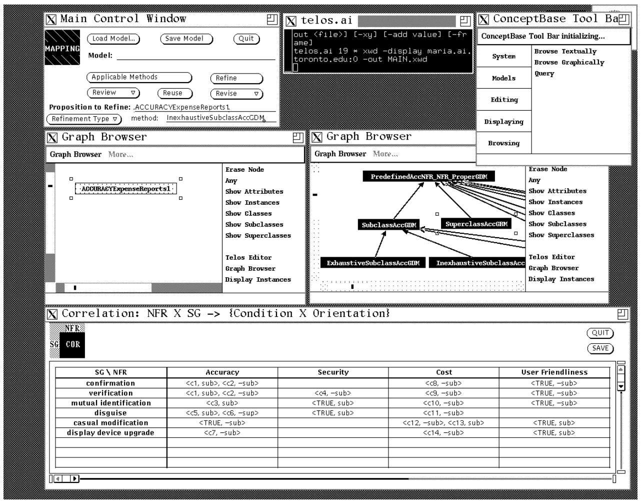


Figure 3: Main control window for the NFR-Assistant.

Correlation rules are intended to formally and systematically represent and assist detection of synergistic or antagonistic interactions between goals. Correlation rules can be used to generate new links between existing goals, as well as to suggest the generation of new goals, hence making tradeoffs explicit. Consider, for example, the developer satisfices confidentiality via an *Authorization* method, which was further refined to an *Authentication* satisficing goal. Now that *Authentication*[`Document`] has an impact (here positive) on *Accuracy*[`Document`], a correlation rule generates a link between these goals. Also a rule may note that fingerprint *(biometric)* authentication improves the security of a system, while hindering user-friendliness (See dotted line at left of Figure 1). The consideration of design tradeoffs are facilitated by the NFR-Assistant's correlation table which shows how an NFR and satisficing goal are related.

[4] In the current NFR-Assistant the method hierarchy has 126 methods for decomposition (of which 23 are implemented); 153 for satisficing (16); and 17 for argumentation (13).

The nature and strengths of the relationships between development decisions is represented by link types, and the effects of each development decision are systematically evaluated. A goal can contribute positively to another goal (designated as a *sub* link), or negatively *(–sub)* as between *Biometric* and *Userfriendliness*. The conflict between these last two goals is resolved in favor of user-friendliness by stating an *argument* that user-friendliness is more important. Throughout the goal graph expansion process, the effect of each design decision is propagated from offspring to parents via a *labelling (evaluation) procedure* which labels notes as *satisficed (S), denied (D), undetermined (U),* etc., to assess the degree of goal achievement.

Both satisficed and denied alternatives form part of the *development history,* the goal graph structure, which uses all its components to systematically and concisely record development actions and their consequences

## 3 Methodology for Using the NFR-Framework

We now describe the process by which one can use the NFR-Framework. This was used in carrying out the experimental studies.

**Step 1: Knowledge acquisition:**

**Step 1.1: Acquisition of knowledge specific to NFRs.** First we obtain knowledge about the particular type of requirement, e.g., security or performance. Our sources of knowledge included industrial experience and academic literature. We encoded this knowledge and catalogued it to produce a terminology for the quality requirements, a list of generic techniques, and their tradeoffs and interactions.

**Step 1.2: Acquisition of domain knowledge.** We obtained documents from (or about) the organisations whose systems we were studying. Depending on the organisation, the nature of the documents varied. In one case, we received system development documents (requirements, design, implementation). In another we obtained partial schema information, workload information, and information on the organisation, its operations, and its system. In yet another, we received only general, public information (e.g., annual reports). We then conducted an initial review of the documents, to understand the domain characteristics, obtain as much NFR-related information as possible, and get an initial idea of some of the main requirements. Due to the varying amounts of information available to us, we sometimes had to provide or augment information on schema and requirements, based on our own experience, but without violating available domain knowledge.

Detailed development via application of the framework involved iterative use of the next two steps.

**Step 2: Application of the NFR-Framework:**
**Step 2.1: Identification of NFR-related concepts.** This Step takes domain information and sorts it into categories along the lines of the main components of the NFR-Framework.

1. *Identification of important NFR goals.* We came up with an initial estimate of what is important, critical, etc., for the system.
2. *Identification of development (design, implementation) techniques.* These include: (a) those described in document; and (b) those from our knowledge of the area of the quality requirement (from Step 1.1)
3. *Identification of design rationale (arguments).* We identified and catalogued the relevant arguments used.

**Step 2.2: Linking NFR-related concepts.** In this Step we use and link the concepts identified in Step 2, in order to show how the framework might be used to meet the stated NFR goals during the development of a system. In conducting these studies, we consider whether the identified concepts "hang together" by seeing whether goal-graph structures can be produced in a meaningful way to meet needs of real systems.

1. *Refining, clarifying and relating goals.*
   (a) We related development (design, implementation) techniques to NFR goals.
   (b) We identified conflict and synergy.
   (c) We clarified the meanings of goals, both *individually* (i.e., when the document suggests a goal which cannot be easily or directly clarified into a known category), and *associatively,* for which there are two cases:
      i. where several goals had direct dependencies among one another (e.g., security and confidentiality).
      ii. where one or more decomposition methods had to be introduced before the relationship between two (or more) given goals could be established. We called our approach "archaeology mode," in which we attempted to "dig" into the documents to identify and fill in gaps, while producing goal graphs without being in contact with the developers.
   (d) We identified factors underlying correlations.
2. *Identification of critical NFR goals.*
   - This was related to identifying what development (design, implementation) techniques were repeatedly emphasised.
   - This also involved identifying critical goals, and critical and dominant parts of the workload.
3. *Providing design rationale.*
   (a) We located applicable design rationale from either the source document, or the workload description.
   (b) On the basis of development tradeoffs present in the goal graph structure under construction, we can provide some design rationale as to why certain particular techniques were chosen.
4. *Assessing goal achievement.* We applied the evaluation procedure to determine whether the stated NFRs, particularly the important ones, were achieved. Part of the procedure involves examination of the nature and strength of relationships between goals.

The reader is referred to the summaries of the three studies in Appendices A, B and C. Sections 4 and 5 evaluate the results of the studies from two viewpoints.

## 4 Our Own Evaluation as Framework Developers

In all three studies on credit cards (described in Appendix A), public health insurance (Appendix B), and government administration (Appendix C for Cabinet and Taxation) information systems, differences in domain characteristics were reflected in the types of NFRs stated, the number of method applications, and the topology of goal graph structures. Based on our own experience as developers and users of the framework, we describe both positive and negative findings.

**Positive Findings.** To deal with NFRs, we feel that there is a need to consider key domain characteristics, capture NFR-specific concepts, and detect defects. Overall, we feel that these three studies suggest that this need is dealt with and, to a certain extent, met by the NFR-Framework.

*Framework observations.* According to Boehm's [Boehm78] insightful experimentation, when the developers are made aware of quality concerns, that by itself helps improve the overall software quality. This was reflected in our experience: the *explicit* expression of an initial set of NFRs as goals improved awareness and commitment, and led to systematic development. Furthermore, when conflicts and synergy among the NFR goals were explicitly described, we were able to consider design tradeoffs in selecting among alternatives to satisfice NFR goals. Given a set of goals, we use a prioritisation technique to determine which goals are most important to satisfice; this need to identify and focus on *critical* goals has been pointed out for quality goals in general [McCabe87] [Juran79] and performance in particular [Smith90]. Also, in the presence of competing alternatives, expressing critical goals and the effect of each design decision (using the labelling facility) helped us resolve conflicts and record design rationale for both accepted and rejected alternatives. With all of this explicitness in place, we were able to *explicitly and concisely* link design decisions back to the source NFRs, which we feel could improve traceability.

*Catalogues of Methods* enabled us to succinctly capture the large number of NFR-specific concepts, such as security and performance, and their associated techniques, in an organised manner. In particular, methods, with their division into decomposition, satisficing, and argumentation, provided us with a needed vocabulary and subject matter during the development process. This, along with sorts and method hierarchies, acted as a basis for thinking about NFRs that were central to our studies, and helped identify factors important for the satisficing of NFR goals. For performance requirements, for example, we were able to capture knowledge of implementation techniques, workload statistics and layers as methods. This knowledge was captured in methods which were further organised into hierarchies along several axes. This was found helpful during the Government Taxation study.

*Goal graph structures* are important as *development history*, both as a record of initial development and for long-term review and maintenance of a system over time. For example, the Health Insurance system had been subject to a lengthy review which proposed long-term changes to the system. We felt that a goal-graph structure much more concisely captured the review's essential descriptions: many NFRs were already given, and our work mainly involved finding the right place for them in a goal graph structure. To focus attention on a portion of a goal graph dealing with a class of issues, the use of a performance-based layering approach was found helpful in the credit card study.

*Defect detection observations.* Dealing with NFRs for even small portions of these systems involved repeated clarification of goals, addition of missing details, detection of goal synergy and conflict. We now describe our experience in using the NFR-Framework's (semi-)formal and systematic approach to detecting faults.

1. Goals were sometimes ambiguous in the documents, mostly in sorts, hence leading to various interpretations. Methods helped clarifying the meanings of particular sorts which varied among applications. For instance, the notion of sort security varied from credit card fraud prevention (a form of commercial integrity [Clark87]), to confidentiality of medical information, to confidentiality of government records. For performance goals, we had to distinguish minimising response time from minimising variability in response time (following [Smith90]) by defining additional specialised sorts.

2. By distinguishing goals into three types, the framework helped us to detect omissions. *Omission of NFR goals* is detected when there are NFR-related satisficing goals without corresponding NFR goals. This makes it hard to detect goal synergy or conflict, and hinders enforcement of appropriate satisficing goals. For the Cabinet Document system, for example, the operating cost goal was implicitly inferred from the documentation on the information collection and maintenance effort. *Omission of satisficing goals* makes the quality of the target design or implementation less than acceptable. In the Credit Card system, direct communication devices help prevent fraud for security, but this raises the need to address wire-tapping, as recorded in a correlation rule. *Omission of arguments* makes it hard to justify why particular decisions are made. For the Health Insurance system,

for example, we were wondering why *only* few internal checks were performed for confidentiality.

3. Antagonistic correlation rules helped detect conflicts, which were then often resolved by decisions in favor of critical goals. For the Health Insurance system, *only after a thorough review* by the investigators involved, was it discovered that the accuracy of claims conflicted with their confidentiality if certain methods were carried out by a third party. If this correlation rule is present, the next evolution of a similar system would take much less effort.[5] The combination of identification of critical goals and negative correlations also arose in the Government Taxation study, where it was used to select an implementation which dealt effectively with a time-space tradeoff.

4. Detection of *redundancies* in the source description was facilitated largely by the use of the framework's semi-formal notation which distinguishes three kinds of goals, their sorts and parameters, and identifies their corresponding methods and correlation rules. For example, for the Health Insurance System, there were several descriptions of why certain mechanisms were good for several objectives of the system; we were cable to capture the mechanisms as satisficing methods and objectives as NFR goals, and without repetition.

5. Synergistic correlation rules and gradual application of satisficing goals helped us to also detect *opportunities.* For instance, we used a rule that when the parameters of two NFR goals are related in a *specialisation (isA)* hierarchy, the application of a satisficing method to the more specialised goal presents an opportunity for applying the same method to the more generalised goal, often without additional cost. For instance, terminals installed at health branch offices for reviewing rejected claims could also be considered for directly transmitting claims in general.

In retrospect, there is a sense in which the use of the framework made defect detection somewhat "syntactic," hence becoming more straightforward. This was due to our use of the framework's semi-formal notation which distinguishes three kinds of goals, their sorts and parameters, and identifies their corresponding methods and correlation rules. For example, the use of a satisficing goal required a corresponding NFR goal. We see this ability to assist defect detection as an important benefit of the framework's approach to formal analysis and systematic software development.

**Negative Findings.** Overall we feel that there is room for improving the framework.

[5] For more on dealing with changes, see [Chung94c].

*Coverage.* Although the coverage (hit ratio) of our methods was high for the current studies, sometimes we had to provide new decomposition methods, to systematically bridge the gap between given NFRs and given satisficing goals. We also felt that the definition and use of more specialised methods would require additional expertise.

*Design Rationale.* We feel that there is a comprehensibility limit, perhaps three or four, on the level of nesting in an argumentative structure. We agree with the view [Yakemovic88] that the deeper the level of argumentation gets, the harder it becomes to understand the meaning of arguments.

*Qualitative Approach.* The framework's qualitative approach allows us to capture our intuitions that are missing in a quantitative metric. However, we observed that sometimes numeric values (e.g., a response time of 1.9 seconds [Visa Canada90], a rating of relative priorities among goals, and the marginal utility of an additional or alternative method application) are available and the framework does not allow systematic treatment of such numeric quantities.

*Scalability.* The goal-graph structures ranged in size from under 100 goals for the individual government systems to over 200 for the credit card system. For fuller-sized studies, we would anticipate the goal graphs to be larger. Although our intuition is that our essential conclusions would remain unchanged, we are not sure whether the process that leads to such conclusions would also remain unchanged.

## 5 Evaluation of Studies by Domain Experts

**Approach.** An important source of evaluation of the framework and our studies comes from those familiar with the domains and systems we studied. To date, we have conducted interviews for the health insurance system (Study 2) with Mr. Dominic Covvey, who is a medical computing consultant and was a consultant to a commission of inquiry on the confidentiality of medical records [Ontario80], and for the government Cabinet documents system (part of Study 3) with Mr. Lou Melli, who is an independent systems consultant, not a part of the government organisation studied, who takes a consultative and interactive approach to dealing with systems and their requirements. Some of the responses seem to generalise in applicability to classes of systems, at least for medical systems and small government systems. We distributed a draft of this paper, a questionnaire, and where appropriate, more detailed versions of the studies (for the two interviews conducted, [Chung93a]).[6] Due to time constraints on the part of the authors, there was not always sufficient time for these interviewees to fully review our material.

### Major Findings.

*NFR-Framework.* The interviewees said that the framework and its components would be helpful for developers, and can be helpful in the broad domains studied (medical and small government systems). The cataloguing of development techniques and NFR-specific knowledge would be helpful, and in some cases is more advanced than current practice in the domains considered; for medical systems, such catalogues are not available to guide people. The goal graph structures and their components were considered helpful. However, improvements in naming and presentation are needed to increase understandability.

*Applicability to Specific Domains Studied.* In conducting the studies, we initially aimed to use the framework to deal with real quality-related problems of real systems. The interviewees felt that our studies were generally based on their domains, and illustrated our approach, but one said that our goal prioritisation and results were different than what he would have produced. We thus realised that our lack of consultation with domain people during the study left gaps in our domain knowledge. The other could not evaluate our coverage since its level of presentation was too shallow compared to the level of detail in the particular system studied (which was old, ill-structured and hard even for experts to understand).

*Applicability to Broader Domains.* There was a warm appreciation of the potential of the Framework for the broader classes of systems considered. One interviewee felt that our techniques could be used in the medical field, such as in developing procurement criteria when enhancing products, both existing ones and those under development. The government consultant was able to relate the framework's decomposition approach to an adaptive interview-based approach used by consultants in determining requirements of a small government system.

*Training Costs and Payoff.* An open question was raised: Will the time and costs needed for training staff in the framework and for developing goal graph structures be surpassed by payoffs during development?

### Framework Findings.

*Goal Graph Structures.* Overall, these were seen as useful. One interviewee understood them, but the other understood the approach, but not the details of the goal graph structure (This was remedied by a presentation of the framework during the interview).

*Framework Emphases.* There was an appreciation of the framework's emphases. Its focus on evaluation was considered important. It was felt that there is a need to clarify requirements and meet requirements.

*Formality.* The ability to be more formal was considered positive. However, the framework would have to be useable to domain and systems staff who currently work much more informally.

*Process-Oriented Approach.* One interviewee questioned the framework's premise that the quality of a product depends on the quality of the process. For example, there can be quality processes which do not lead to quality products.

*Tradeoffs (Correlations).* It was felt that it is good to deal with tradeoffs early. Currently, there is not a balanced treatment of tradeoffs for some medical systems: they are secure, but rigid; to increase flexibility, security features may be disabled. On the other hand, for small government systems, a consultant can record conflict and synergy by using interviews and developing prototypes, which are then discussed with the client.

*Determining Main Requirements.* One open question is how one finds the main (top-level) requirements. Furthermore, can goals be dealt with in small clusters, and the results then combined?

### Application Domain Findings.

*Domain Knowledge and Proportionality.* In the interview, we learned that our Cabinet study did not always correspond to the domain because we lacked some knowledge about the domain, its priorities and terminology; we attribute this to our lack of contact with domain people during the study. We treated security as a main goal, but in the domain, accuracy actually predominated other NFRs.[7] Similarly, we detected an interaction, but it was among requirements

[6] Both interviewees have been associated with the University of Toronto. For the other studies, credit card and government taxation systems, we have made contacts and expect to soon conduct interviews.

[7] Another reason why security was less crucial than we thought, was that clients in the particular domain seemed primarily interested in bringing the system into existence. In addition, the system would not store the actual Cabinet documents, just "header information" (such as topic and date).

which were minor in the domain. We also found that some of our arguments about the domain were not quite right, again due to our lack of knowledge about the domain and its terminology.

*Usage of the Framework.* Both interviewees pointed out that processes and methodologies need people to be used in practice. A variety of people (developers, administrators, etc.) would have to be trained in the use the framework.

**Lessons Learned.**

Comments from domain experts are valuable "reality checks" on our work. In reviewing the interviews, we feel that future studies could benefit from a more interactive and consultative approach, which would involve repeated contact early in the study to ensure that we have appropriately captured the priorities for requirements; this might allow studies to be closer to actual problems in the organisations studied, and allow more specific evaluation of the results of studies.

## 6 Related Work

Our focus has been on evaluating one framework by way of three studies of real systems which treat several NFRs. In contrast, Finkelstein and Green [Finkelstein93] compare several goal-oriented approaches, including the NFR-Framework, to treat a set of general requirements for one example. There are other uses of the NFR-Framework, including organization modelling [YuForthcoming], and project risk management [Parmakson93].

Other fuller studies include performance for a hospital system [Brataas92] using a quantitative approach, and accuracy for a a criminal-record system [Laudon86] using domain analysis. Our studies dealt with several NFRs using a framework which advocates a systematic qualitative approach.

*Quality Function Deployment* (or *the House of Quality)* [Hauser88], in industrial engineering, has been applied to Software Quality Assurance (e.g., [Zultner92] [McCabe87]). Both QFD and the NFR-Framework can be used as media for communication and planning and to provide a conceptual map from customers' requirements to designs and implementations. However, the NFR-Framework focusses on a (semi-formal) representation and systematic development process, with the additional features of design rationale. While many studies (e.g., [Sullivan86] [Kogure83]) are available of the effectiveness of QFD in improving industrial quality, our present paper is one of the first to report experience in using the NFR-Framework for several software systems. It would be interesting to examine whether the NFR-Framework reduces rework and scrap, hence inducing shorter production time and lower cost, like its sibling QFD.

Thorough evaluations have been reported of some quantitative, more product-oriented approaches which have an emphasis on defect defection and reduction. For instance, [Linger93] gives an elegant evaluation of the cleanroom approach whose emphasis is on carrying out different inspection tasks by independent teams. Similarly, [Schneider92] gives a realistic evaluation of the *N-fold inspection technique* whose emphasis is on replicating the same requirements inspection task by independent teams. This technique is noted for inspecting written general requirements primarily for detecting defects. Unlike these evaluations, our studies focus on NFRs and evaluate a qualitative, process-oriented, generative framework which takes a (semi-formal) and systematic approach to representing and using NFRs during software development.

Related to our observations on defect detection is a large empirical study [Gotel94] of requirements *traceability*, which involves developing and following the life of a requirement from its origin through all phases of development.

## 7 Conclusions

We have conducted three studies of small portions of software systems, in order to give an initial evaluation of a recent framework for dealing with non-functional requirements (NFRs) in the software development process. The studies dealt with several NFRs, primarily accuracy, security and performance.

From our perspectives as framework developers and users, we feel that the NFR-Framework helped us to *represent* and *use* the large number of NFR-specific concepts that were central to our studies, in an organised manner. The representations provided us with a needed *subject matter* and *vocabulary*, which acted as a basis for thinking about a variety of NFRs. Using the representations, then, we were able to consider design alternatives and their tradeoffs with respect to conflicts and synergy among the NFRs, to determine the effect of each design decision, and to justify the decision with the needed design rationale for both accepted and rejected alternatives. With this ability, we were able to *explicitly and concisely* link design decisions back to the source NFRs. Using the representations, which distinguishes three kinds of goals, their sorts and parameters, and identifies their corresponding methods and correlation rules, we were also able to *detect defects*. Throughout the process, we feel that goal graph structure is important not only as a means for developing quality software but as development history.

From the perspective of the domain experts we have consulted to date, the NFR-Framework would be helpful for developers. The cataloguing of development techniques and NFR-specific knowledge would be helpful, and in some cases is more advanced than current practice in the domains considered. Our studies were based on their domains, and illustrated our approach. However the detailed conclusions in the studies are not necessarily what would have been reached by the organisations studied; this seems mainly due to our use of source documents without ongoing contact with domain experts during our study. Nonetheless, there was significant appreciation of the prospects for applying the framework in the broader fields studied, particularly medical computing. We are arranging interviews with experts from the other domains we studied.

We and the domain experts have identified directions for future work. Hence, we feel that our studies should be regarded only as an initial evaluation. We have learned the importance of capturing domain expertise and feedback, preferably early in the study process, in order to have the study to correspond more closely to the needs of the organisation studied. The *scalability* of the framework with respect to larger studies is an open issue. Furthermore, it is not clear that significantly larger studies could be handled by the NFR-Assistant tool as currently implemented, due to its slowness when used for the present studies. We also need to find out a (cost-)effective way to train a variety of users (developers, administrators, etc.) in the use of the framework. To resolve these issues, larger and more diversified applications are needed during an *actual* system development, as opposed to an "off-line" study. This will help determine more fully the usefulness of the framework, and the validity of its premise that the quality of the software product depends on the quality of the process used to develop it.

Notwithstanding the open issues of the framework, we feel that the evaluation of the studies has shown some preliminary evidence for the contribution of the NFR-Framework to the development of quality systems.

**Acknowledgements:** Our sincere gratitude to Professor John Mylopoulos and Eric Yu for moral support, encouragement and ongoing discussions about this work. Eric made excellent suggestions in graciously reading several recent drafts. The NFR-Assistant was implemented with assistance and advice from Ian Maione, Thomas Rose and David Lauzon. Many thanks to Fernand Livernoche and G. W. Venner, of the Dept. of National Revenue, Taxation, for providing information on the income tax appeals process. We thank Lou Melli and Dominic Covvey for their review of a draft of this paper and insightful comments and clarifications, and for sharing their enthusiasm and domain expertise with us. We also thank Honey Robb of National Revenue and Patrick Daly of the Canadian Bankers' Association for information and referrals which we trust will lead to fruitful interviews about their domains.

## A Study 1: Credit Card System

**Domain characteristics.** We consider an information system for a bank's credit card operation, which maintains information on sales, cardholders and merchants. Transactions are authorised, and accounts are updated. From the actual statistics for use of Visa and MasterCard credit cards for all banks in Canada for the year ended 31 October 1991 [Canadian Bankers91], we estimate workload for an hypothetical large bank, assuming it has 15% of the national market. It would have 3 600 000 cards in circulation and 90 000 merchants; each day it would deal with 217 000 sales slips, 218 lost or stolen cards, and 19 cards used fraudulently.

**Treatment of NFRs.** We examined [Chung93b] [Nixon93] some organizations' documents [Canadian Bankers91] [Visa Canada90] [Visa International91] [MasterCard91] which show differences in domain characteristics as well as in the types of NFRs. For a commercial credit card system, quality is very important, as the market is highly competitive. Timeliness, security and response time were considered important quality factors. Their important synergy for the case of reducing the time between a transaction and its posting to an account is pointed out in the document:

> The benefits of this time reduction are enormous: we are able to to post more rapidly to the cardholder account, improve risk control by lowering exposure to transaction in the pipeline, and reduce float costs for both merchants and merchant Members. [Visa Canada90]

The informal concerns for security are sometimes *ambiguous,* perhaps due to the nature of the document, inviting many possible interpretations. For example, an NFR or satisficing goal needs clarification when it is expressed only in terms of its sort name without clear explanation. Recognizing the importance of accuracy and confidentiality aspects of security, we decomposed a security goal, `Sec[Account]` (top of Figure 4, taken from [Chung93b]), on its sort into goals for confidentiality, availability and integrity. The integrity goal was further decomposed into accuracy and completeness goals.[8]

Further refinements led to the goal `TimelyAccuracy[Transaction]`, the faithful representation of information during the lifetime of the transaction. Using a prioritisation technique, this goal was treated as critical. This was supported by market surveys show that timely accuracy is of strategic importance, recorded in an argument adapted from the *"vital few — trivial many"* principle (the "80–20 rule") [Juran79] [McCabe87].

Other methods are also applied to reflect that a security breach could take place either internally, by staff accessing the system, or externally, in terms of forgery of vouchers, remittance requests, etc. There is a trade-off between the benefits of installing the device (`InstallInputDevice[Transaction]`) and the increased costs (`EquipmentCost[Transmission`). The system used correlation rules to detect this potential conflict. Here, we selected the input device by arguing that the synergistic benefits outweighed the antagonistic penalties. The time to transmit transaction information to the system was decreased by installing direct communication devices, *synergistically* improving timeliness and security, which the company used as a major selling point. But this method can bring negative side effects, such as security breaches due to wire tapping. The framework can assist detecting and reporting such omissions. One result of this design process is that input devices can be installed at the transaction point to improve security of cardholder accounts.

Performance is another important NFR due to the high transaction volume, and the need to cancel stolen cards immediately. Good time performance means seeking fast response, while good space performance means low usage. We repeatedly followed a pattern of first decomposing a goal based on its components,

[8] This notion of accuracy seems to be similar to that in [Heninger80] *external consistency* [ITSEC91], comparable to the notion of *authenticity* [Parker91], and to the notion of *validity*, which is complementary to *completeness* for database integrity [Motro89].

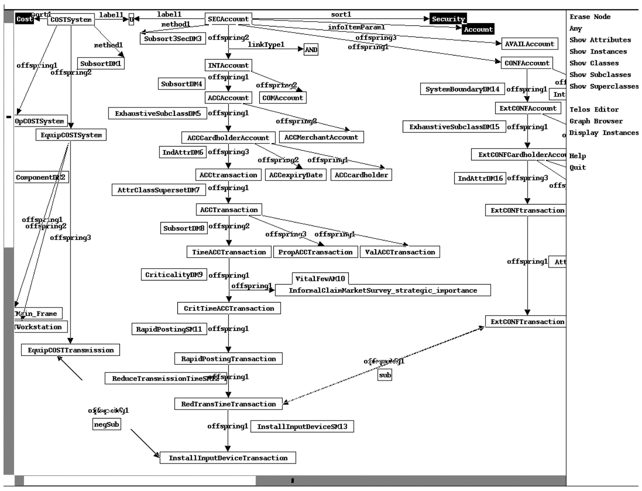


Figure 4: A Security goal graph structure for credit card system.

and then identifying and focussing on *critical* component goals, in keeping with principles for achieving *responsiveness* [Smith90]. We decomposed a goal of good response time for cancelling lost and stolen cards into goals for the operational components. We focussed first on accessing the attributes of a credit card, then on its status attribute, and finally on updating it, which is critical for cancelling a card. This was satisficed by updating status at the *start* of the operation, which effectively cancels the card, even though the non-critical tasks may continue in the background.

Space performance goals sometimes interacted negatively with the above treatment of time goals. For example, we sought good space performance for storing the attributes of a card. As there are many cards, using a compressed format would save space, but compressing the status attribute would cause a conflict by slowing down the critical operation of retrieving a card's status. So we satisficed the space goal for the dominant case, by compressing the other attributes. We avoided a negative impact on the critical time goal by using an uncompressed format for status, which (acceptably) did not meet a non-dominant storage goal.

# B Study 2: Health Insurance System

**Domain characteristics.** A health insurance system [Ontario80] involves governmental, professional and some commercial organizations. Here, confidentiality is a very important concern due to the sensitive nature of medical treatment information, and accuracy of rejected claims is also important. Most information on patients' treatments and practitioners' claims is taken from the claims cards. Reports of breaches of confidentiality brought about a major investigation, which reported that breaches of confidentiality were likely to occur externally to the system, involving service bureaus and couriers which handle patients' personal information.

**Treatment of NFRs.** In refining a goal for confidentiality of treatment information, we were able to focus on the aspect of *external confidentiality,* which the investigators emphasised. Among the many government health offices, the investigators identified a small number of critical branches from a larger number. Accordingly, we narrowed the focus in our study, first by considering individual offices on a branch-by-branch basis using the *compartmentalisation* method, and then by identifying *critical* branches using a prioritisation technique adapted from the *"vital few — trivial many"* principle [Juran79] [McCabe87], which focusses attention on the few items which are vital. This reduction of the search space, in accordance with the investigators' report, provided evidence for the usefulness of the *"vital few — trivial many"* argumentation method. Similarly, in handling the accuracy of claim information, we were able to gradually increase the criticality for certain categories of rejected claims.

One important discovery made by the investigators was that service bureaus can enforce certain methods (e.g., verification and consistency checking) which work synergistically, hence improving accuracy, boosting employee morale, and decreasing operating costs for claims processing. This synergy was later expressed more precisely by introducing appropriate sorts and parameters for the three goals: *Accuracy*[`LargeRejectedClaim; veryCritical`], *User-friendliness*[`ClaimsInputActivity`], and *OperatingCost*[`manpower`].

However, these methods are in conflict with a confidentiality goal for the treatment information handled by the district branches, since the service bureau will be acting on behalf of the practitioner:

> The new procedure ... appears to have several benefits including higher employee morale and fewer claim rejections. A more important benefit will be the reduced paper requirement due to the input of claims information on magnetic media. However, the new system introduces a third party to the process. [Ontario80] (p. 151, emphasis in the original)

We were able to concisely represent the above situations in a goal graph structure (Figure 5), including the detected correlations; using a service bureau counter-acts the *ReduceIntermediateAgent* method, which provides a shorter transmission path that enhances confidentiality. As design rationale is quite complex, many other correlations are not shown in the figure; for example, the *UserFriendliness* and *OperatingCost* goals are correlated with *VerifiedBy,* as well as with other satisficing methods. Nonetheless, even the simplified goal graph structure reflects the "third-party" issue by showing the trade-offs between verification by internal staff vs. by an external service bureau.

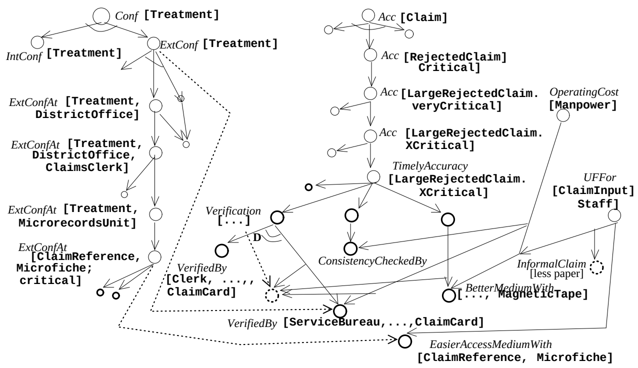


Figure 5: A goal graph structure for Health Insurance System.

## C Study 3: Government Administration Systems

**Domain characteristics.** We now consider a class of government information systems which provide support for decision-making over a long period of time, recording and tracking information and decisions. We consider projected extensions to two instances of this class of systems: (1) a Cabinet Document system [Chung93a], which is used by a government ministry to manage the flow of documents about the decision-making process of the Cabinet, and (2) an Income Tax Appeals system [Nixon94], which manages the handling of taxpayers' appeals of decisions about their tax returns. Like many other government systems, both manage long-term processes, which are subject to constraints, and last months or even years. Both are highly consultative and interactive: blockage somewhere in an decision-making process can stop progress altogether, while one event can trigger a variety of other events.

The projected Cabinet Document system[9] manages documents submitted for Cabinet consideration, as well as their schedules and history throughout the approval process. The process requires concurrence of several committees and Ministries. There are various types of Cabinet documents whose estimated volumes of transactions are low. For instance, 500 Cabinet submissions are expected per year, each requesting approval by Cabinet. The initial system had been partially automated using spread-sheets, and we considered a projected extension.

The Income Tax Appeals system manages the handling of appeals by taxpayers who disagree with decisions made by the Canadian Department of National Revenue. During the appeals process, a taxation officer should contact the taxpayer, and may also consult other staff members. While a system specification of the existing appeals database was not available to us, we had access to some statistics and operational procedure manuals [Revenue Canada80,89,91,92a,b,c], which partially describe the appeals system. Actual taxation workload statistics are used in this study: in a recent year, 57 072 appeals were completed, and between 18% and 55% of decisions were made within 30 days, well in advance of the service-standard deadline. Currently, workload control and the supervision of the progress of appeals cases are not automated. As a partially automated tracking and follow-up service would be helpful,[10] we have added design specifications and performance requirements which could reasonably be expected for such a system, including the issuing of reminders to staff to consult taxpayers before deadlines.

**NFR Considerations.** Our governmental studies primarily addressed confidentiality, user-friendliness, and performance. *Confidentiality* is important for government systems in general, and was explicitly expressed in terms of information items and those staff members who are allowed to access them.[11] The system manages information about documents at the Cabinet level: the number of documents is fairly small, but each is important. *User-friendliness* requirements explicitly state that simple and helpful interfaces should be provided. *Performance* requirements are quite important for government systems, such as the Taxation system, which has a number of appeals to process over a period of time.

**Treatment of NFRs.** *Cabinet Document System.* The Cabinet Document system, which has a relatively low transaction volume, deals solely with the flow of information within government. This was reflected in the types of decompositions, as well as satisficing methods. We focussed on decomposing *confidentiality* goals, their parameters and satisficing methods which are suitable for the well-defined goals. Consistent with the source document, we also decomposed the user-friendliness goal into goals with sorts simplicity and helpfulness.

The Cabinet Submissions have several attributes: many attributes may be examined by some staff, but the attribute relating to one kind of document, Management Board submissions, may not be shown to certain staff members. Furthermore, this attribute may either be derived or explicitly input. We handled this situation by first using the *IndividualAttributes* decomposition method. Then, in choosing a representation, there were many implicit and explicit correlations, reflecting tradeoffs among confidentiality, accuracy and user-friendliness goals. Reducing exposure of the Management Board attribute for security purposes is *implicitly* in conflict with allowing user-friendly input menus, while the *crossReference* method for accuracy can *explicitly* make operating costs exceed the budget limit, since it requires additional operations on information items.

Of the security considerations, the confidentiality aspect is stated as requirements, while concerns for accuracy are evident as design-level procedures (satisficing methods). As our study was based only on the source documents, we used the goals and and arguments to try to determine the accuracy NFR goals which were implicit. For example, to clarify the purpose of various types of *consistencyChecking* methods described in the source document which seemed quite appropriate and carefully chosen, we introduced accuracy goals, stated the methods as effective, and evaluated the goals via the labelling procedure as satisficed. To satisfice confidentiality goals, it appears that a *passwordAuthentication* mechanism suffices. We could use argumentation methods to attach the likely design rationale.

*Tax Appeals System.* We started with a goal of good *time* performance for issuing reminders to appeals staff to contact taxpayers (Figure 6, adapted

[9] The details of this example are based on private communications with Lou Melli. He reported that the development team explored alternatives.

[10] Letter from Fernand Livernoche, Acting Director, Policy and Programs Division, Department of National Revenue, Taxation, November 19, 1992.

[11] For the purposes of the Cabinet documents study, we treated confidentiality as the main requirement; we subsequently learned from a domain expert (Section 5) that accuracy was considered paramount.

from [Nixon94]). Knowledge of the many implemen-

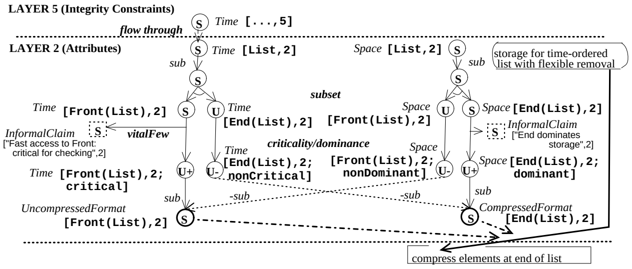


Figure 6: A Performance Goal Graph Structure for Government Tax Appeals.

tation techniques for semantic data models [Hull87] [Nixon90], here particularly temporal integrity constraints (See [Chung88]), was represented and used in conjunction with knowledge of the actual application and its workload statistics. Arguments based on actual statistics and workload patterns (e.g., many cases are resolved faster than the service standard requires) were used to rule out the use of some implementation techniques. As a result, an appropriate implementation structure was selected, consisting of a list of actions ordered by the time by which each had to be executed; the list allowed early removal of any entry, which facilitated handling cases completed before the deadline.

Having dealt above with integrity constraints, we then focussed on attributes of the list. Here we had to deal with conflicting time and space goals. For responsiveness [Smith90], we focussed on critical and dominant parts of the workload, and also used knowledge of implementation techniques and the actual workload: fast access to one portion of the list (the front) was noted as being critical for checking constraints, while the remainer of the list (whose size is dependent on the number of tax appeals: 38 532 at a recent year-end) was noted as dominating storage. The result provided good time performance for the critical part of the list, and good space performance for the large remainder of the list. By satisficing the critical and dominant goals, while denying the non-critical and non-dominant ones, the happy result was that both the overall time and space goals for the list were considered satisficed, even though at first glance they seemed to be unreconcilably in conflict.

In generating implementations from designs, we used a *layering* approach [Nixon91] which showed where to introduce domain knowledge associated with design components. This helped control the number of potentially interacting implementation concepts to consider at a time. For example, we used a goal graph at a higher layer to deal with integrity constraints, and a lower one to deal with attributes. This approach was inspired by a layered framework for performance prediction for relational databases [Hyslop91].

## Bibliography


**[Benzel89]** T. C. Vickers Benzel, "Developing Trusted Systems Using DOD–STD–2167A," *5th Annual Computer Security Applications Conf.*, Tucson, Arizona, Dec. 4–8, 1989, pp. 166–176.

**[Boehm78]** B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. J. MacLeod and M. J. Merritt, *Characteristics of Software Quality.* Amsterdam: North-Holland, 1978.

**[Boehm87]** B. Boehm, "Industrial Software Metrics Top Ten List", *IEEE Software*, Sept. 1987.

**[Brataas92]** Gunnar Brataas, Andreas L. Opdahl, Vidar Vetland and Arne Sølvberg, *Information Systems: Final Evaluation of the IMSE.* Technical Report, IMSE Project Deliverable D6.6–2, SINTEF (Univ. of Trondheim), Norway, Feb. 27, 1992.

**[Canadian Bankers91]** Canadian Bankers' Association, *MasterCard and Visa Statistics*, Toronto, Ont., Dec. 1991.

**[Chung88]** K. Lawrence Chung, Daniel Rios-Zertuche, Brian A. Nixon and John Mylopoulos, "Process Management and Assertion Enforcement for a Semantic Data Model." In J. W. Schmidt, S. Ceri and M. Missikof (Editors), *Advances in Database Technology — EDBT '88,* Int. Conf. on Extending Database Technology, Venice, Italy, Mar. 1988, Proc. Lecture Notes in Computer Science, No. 303. Berlin: Springer-Verlag, 1988, pp. 469–487.

**[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, Brian A. Nixon and Eric Yu, "Using Non-Functional Requirements to Systematically Support Change." submitted for publication, 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.

**[Dardenne91]** A. Dardenne, A. van Lamsweerde and S. Fickas, "Goal-directed Requirements Acquisition," *Science of Computer Programming,* Special issue on *6th Int. Workshop on Software Specification and Design,* Como, Italy, 1991.

**[Feather93]** Martin S. Feather, Requirements Reconnoitering at the Juncture of Domain and Instance. *Proc. of the IEEE Int. Symp. on Requirements Eng.,* San Diego, CA, January 4–6, 1993. Los Alamitos, CA: IEEE Computer Society Press, pp. 73–76.

**[Fickas87]** Stephen F. Fickas, *Automating the Software Specification Process*, Technical Report 87–05, Computer Science Dept. Univ. of Oregon, Dec. 1987.

**[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.

**[Hahn91]** U. Hahn, M. Jarke and T. Rose, "Teamwork Support in a Knowledge-Based Information Systems Environment," *IEEE Trans. Software Eng.,* vol. 17, no. 5, May 1991, pp. 467–482,

**[Hauser88]** J. R. Hauser and D. Clausing, "The House of Quality," *Harvard Business Review,* May–June 1988, pp. 63–73.

**[Heninger80]** K. L. Heninger, "Specifying Software Requirements for Complex Systems: New Techniques and Their Application," *IEEE Trans. on Software Eng.*, vol. SE–6, no. 1, Jan. 1980, pp. 2–13, Reprinted in R. H. Thayer and M. Dorfman (eds.), *Tutorial: System and Software Requirements Engineering,* IEEE Computer Society Press, 1990.

**[Hull87]** Richard Hull and Roger King, "Semantic Database Modeling: Survey, Applications, and Research Issues." *Computing Surveys,* Vol. 19, No. 3, Sept. 1987, pp. 201–260.

**[Hyslop91]** William F. Hyslop, *Performance Prediction of Relational Database Management Systems.* Ph.D. Thesis, Dept. of Computer Science, Univ. of Toronto, 1991.

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

**[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.

**[Jarke93]** Matthias Jarke, Janis Bubenko, Colette Rolland, Alistair Sutcliffe, and Yannis Vassiliou, "Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis", *Int. Symposium on Requirements Engineering,* San Diego, Ca., Jan. 4–6, 1993.

**[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.

**[Kogure83]** Massao Kogure and Yoji Akao, Quality Function Deployment and CWQC in Japan. *Quality Progress,* October 1983, pp. 25–29.

**[Laudon86]** Kenneth C. Laudon, "Data Quality and Due Process in Large Interorganizational Record Systems," *Communications of the ACM,* vol. 29, no. 1, Jan. 1986, pp. 4–11.

**[ELee92]** E. S. Lee, P. I. P. Boulton, B. W. Thomson, and R. E. Soper, *Composable Trusted Systems,* Tech. Report CSRI–272, Computer Systems Research Institute, Univ. of Toronto, May 31, 1992.

**[JLee91]** 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.

**[Linger93]** Richard C. Linger, "Cleanroom Software Engineering for Zero-Defect Software." In *Proc., 15th Int. Conf. on Software Eng.,* Baltimore, MD, May 1993, pp. 2–13.

**[Martin73]** James Martin, *Security, Accuracy, and Privacy in Computer Systems.* Englewood Cliffs, New Jersey: Prentice-Hall, 1973.

**[MasterCard91]** *MasterCard International Annual Report, 1990.* 1991.

**[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.

**[Motro89]** A. Motro, "Integrity = Validity + Completeness", *ACM Trans. Database Sys.*, vol. 14, no. 4, 1989, pp. 480–502.

**[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.

**[Nixon90]** Brian Nixon and John Mylopoulos, "Integration Issues in Implementing Semantic Data Models." In François Bancilhon and Peter Buneman (Eds.), *Advances in Database Programming Languages.* New York: ACM Press, and Reading, MA: Addison-Wesley, 1990, pp. 187–217.

**[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.

**[Ontario80]** "The Ontario Health Insurance Plan Computer System," Chapter 17, Volume II. In *Report on the Commission of Inquiry into the Confidentiality of Health Information,* Vol. I, II, III. Toronto: J. C. Thatcher, Queen's Printer for Ontario, September 30, 1980.

**[Parker91]** Donn B. Parker, "Restating the Foundation of Information Security," *2nd Annual North American Information System Security Symposium,* Oct. 21–23, Toronto, 1991.

**[Parmakson93]** Priit Parmakson, *Representation of Project Risk Management Knowledge.* M.Sc. Thesis, Institute of Informatics, Tallinn Technical Univ., Tallinn, Estonia, 1993.

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

**[Revenue Canada80]** Revenue Canada, Taxation, *Objections and Appeals,* Information Circular 80–7. Ottawa, June 30, 1980.

**[Revenue Canada89]** Revenue Canada, Taxation, *Inside Taxation.* Ottawa, 1989.

**[Revenue Canada91]** Revenue Canada, Taxation, *1991 Taxation Statistics,* Analyzing 1989 T1 Individual Tax Returns and Miscellaneous Statistics. Ottawa, 1991.

**[Revenue Canada92a]** Revenue Canada, Taxation, *Taxation Operations Manual,* Vol. 70: Objections and Appeals. Ottawa, 1992.

**[Revenue Canada92b]** Dept. of National Revenue, Taxation, Appeals Branch, *Quarterly Statistical Report for the Period Ended March 31, 1992.* Ottawa, 1992. Also reports for the (quarterly) periods ended: June 2, 1991; September 27, 1991; and Jan. 3, 1992.

**[Revenue Canada92c]** Dept. of National Revenue, Taxation, Appeals Branch, *Number of days to reassesss Notices of Objection for reassessments between January 1, 1992 and March 1992.* Ottawa, 1992. Also reports for the periods between: Oct. 1, 1991 and Dec. 31, 1991; Apr. 1, 1992 and June 26, 1992; and June 27, 1992 and Sept. 25, 1992.

**[Schneider92]** G. Michael Schneider, Johnny Martin, and W. T. Tsai, "An Experimental Study of Fault Detection in User Requirements Documents", *ACM Trans. on Software Eng. and Methodlogoy,* vol. 1, no. 2, Apr. 1992, pp. 188–204.

**[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.

**[Sullivan86]** L.P. Sullivan, Quality Function Deployment. *Quality Progress,* June 1986, pp. 39–50.

**[Visa Canada90]** Visa Canada Association, *Visa Canada 1990 Regional Report.* Toronto, 1990.

**[Visa International91]** Visa International, *1990 Annual Report*, Canada Region, 1991

**[Yakemovic90]** K. C. B. Yakemovic and J. Conklin, "Observations on a Commercial Use of an Issue-Based Information System", *Proc. of Computer Supported Cooperative Work,* 1990, pp. 105–118.

**[Yu94]** 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.

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

**[Zultner92]** Richard E. Zultner, "Quality Function Deployment (QFD) for Software: Structured Requirements Exploration," In G. Gordon Schulmeyer and James I. McManus (Eds.) *Total Quality Management for Software,* New York: Van Nostrand Reinhold, 1992, pp. 297–317.