Fourth International Conference on Software Quality
McLean, VA, U.S.A. October 3–5, 1994

USING QUALITY REQUIREMENTS TO SYSTEMATICALLY DEVELOP QUALITY SOFTWARE*

Lawrence Chung, Brian A. Nixon and Eric Yu
Department of Computer Science, University of Toronto
Toronto, Ontario, Canada M5S 1A4
{chung,nixon,eric}@ai.toronto.edu, Facsimile (416) 978–1455

Abstract. Although quality issues such as accuracy, security, and performance are often crucial to the success of a software system, there has been no systematic way to achieve quality requirements during system development. We offer a framework and an implemented tool which treat quality requirements as *goals* to be achieved *systematically* during the system development process. We illustrate the process that a developer would go through, in building quality into a system. We have tested the framework on a number of studies involving a variety of quality requirements, organisational settings, and system types.

Keywords: non-functional requirements, accuracy, security, performance, information systems, process, software quality, defect detection, conflicts.

1 Problem

Software development is traditionally driven by functional requirements, i.e., the desired functionality of the system. For example, a credit card system should debit and credit accounts, check credit limits, charge interest, issue monthly statements, and so forth. However, non-functional requirements, such as accuracy, security, and performance, are often just as crucial to the success of the system as the functional requirements. Inaccurate credit account information can lead to monetary loss and damage to reputation of the associated enterprise, while poor response time could lead to loss of customers.

Although the importance of these non-functional, quality issues are widely recognized, there has been no systematic way to build quality into the software as the software is being developed. The main difficulty is to come to grips with the essential concepts of the particular quality, say security, and then, to build guidelines for applying the concepts. Another central difficulty is that each decision made during the development process typically affects many quality issues. For instance, each decision to select a technique to achieve security may also affect other security decisions, as well as performance, accuracy, operating costs, user-friendliness, and maintainability. In attempting to systematically address one type of requirements, say, security, it is hard to be systematic in meeting all the other requirements at the same time. This problem is exacerbated by the complexity of the functionality in most large systems, and by frequent changes to requirements.

What is needed is a systematic framework, and supporting tools, which can keep track of all relevant quality requirements for each development decision, help search for applicable techniques for addressing each type of quality requirements, identify interactions among requirements, assist in evaluating alternatives and making trade-offs, detect defects, and record justifications for decisions, so that the entire

*This paper is largely based on a paper [Chung94] presented at the Workshop on Research Issues in the Intersection Between Software Engineering and Artificial Intelligence. Current affiliation of first author: Computer Science Program, Erik Jonsson School of Engineering and Computer Science, The University of Texas at Dallas, P.O. Box 830688, Richardson, TX 75083–0688, U.S.A.

development process is rational, traceable, and easily revisable.

2 Solution

In our approach, we treat Non-Functional Requirements (NFRs or quality requirements) as *goals* to be addressed during the development process. Making use of well-established methods of addressing each class of goals (e.g., accuracy, security, performance, etc.), the myriad of decisions comprising the development process becomes goal-driven, coherent, and explainable. Incorporation of these concepts naturally led to our process-oriented *NFR-Framework* (See [Mylopoulos92, Chung93a] for details), which is implemented in the form of a tool called the *NFR-Assistant*. The NFR-Assistant provides support for:

1. *Refining initial high-level goals to detailed concrete goals.* The Assistant helps the developer search and select from a catalogue of relevant techniques for addressing the goal under consideration (e.g., authenticate signature to achieve security of credit card transaction).

2. *Identifying the need for tradeoffs.* The Assistant detects synergistic and antagonistic interactions among goals, by invoking pre-defined rules (e.g., authentication by secondary identification improves security, but hinders user-friendliness).

3. *Evaluating and choosing among alternatives.* The Assistant keeps track of the positive and negative impacts of development decisions with respect to all relevant goals.

4. *Recording arguments for or against particular development decisions and tradeoffs.* The Assistant maintains design rationales.

5. *Detecting and correcting omissions, ambiguities, conflicts, and redundancies.* The formal representation and knowledge structuring mechanisms underlying the framework allows the Assistant to alert the developer of defects at each step in the development process.

The Assistant interacts with the developer through a graphical interface. Throughout the development process, the developer is in control. The Assistant manages the details of the process, drawing on a potentially vast base of generic and case-specific knowledge, but brings to the developer's attention only those aspects pertinent to the developer's current focus. This allows the developer to make informed decisions while maintaining perspective. By following this goal-oriented approach, the quality of the software product is assured because quality requirements are brought to bear on development decisions at each step in the process.

The framework and tool were developed by adapting the following artificial intelligence techniques:

1. *a rational design process* [Simon81]: using concepts of goals, means-ends relationships, alternatives, and satisfactory ("good enough") solutions;

2. *problem solving techniques*: in the spirit of AND/OR goal trees [Nilsson71], but augmented with a richer set of goal types and relationships;

3. *reasoning*: from truth maintenance systems augmented with a dialectical style of reasoning from work on design rationale [Lee91], and with qualitative reasoning techniques [AI84];

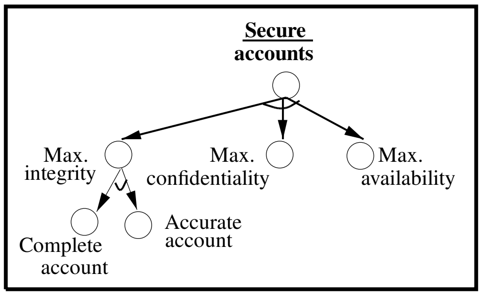
4. *knowledge structuring and modelling features*: from knowledge representation [Mylopoulos91]; and

5. *knowledge base management facilities*: from the ConceptBase facility [Jarke92b] for the NFR-Assistant.

3 Example

To illustrate the use of the NFR-Framework, we show a sample process that a developer of a credit card system would go through. The developer is aided by the *NFR-Assistant.*

3.1 Refining Non-Functional Requirements goals into less-ambiguous sub-goals.



Developer states top security goal
—> NFR-Assistant displays relevant catalogue of disambiguating methods.
Developer selects method
—> NFR-Assistant creates and links sub-goals.

Figure 1(a). Refining Non-Functional Requirements goals into less-ambiguous sub-goals.

Developer states the non-functional requirement *Accounts should be secure*, which is represented by the top goal in Figure 1(a). In the figure, circles denote goals, and arcs denote relationships between goals. This initial non-functional requirement is abstract. On the one hand, it leads to different interpretations for different groups of people; on the other hand, it is coarse-grained and does not permit the consideration of design decisions which normally require more specific details about the requirements.

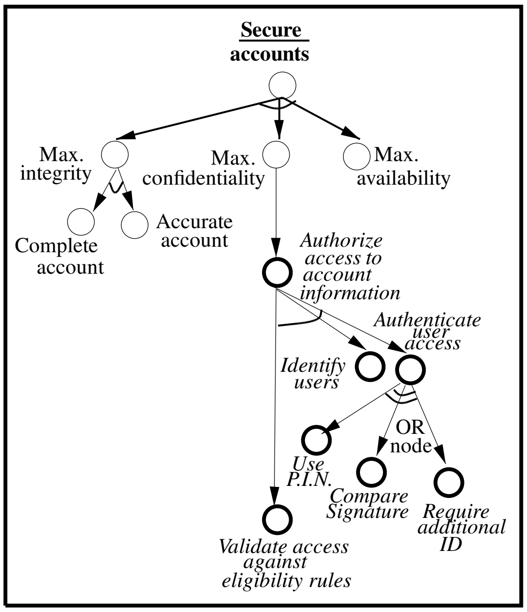
Assistant provides a catalogue of alternatives that are frequently used, helping the developer focus search and be more specific about the security aspect of the requirement.

Developer examines the catalogue. The developer can either select a method from the catalogue or come up with a new one. Here, the developer selects a method from the catalogue which takes the security goal and produces three sub-goals, for Integrity (guarding against unauthorized update or tampering), Confidentiality (guarding against unauthorized disclosure), and Availability (guarding against interruption of service) of the account.

Assistant generates subgoals and links them to the security goal.

This way, the developer successively generates more specific goals to meet the parent goal.

3.2 Choosing among alternative techniques to meet Confidentiality Requirement.



Developer focusses on Confidentiality
—> NFR-Assistant displays catalogues of security assurance techniques and trade-off.
Developer selects technique
—> NFR-Assistant creates and links techniques
—> (see Figure 1(c)).

Figure 1(b). Choosing among alternative techniques to meet the Confidentiality Requirement.

Developer decides to focus on the Confidentiality Requirement in moving towards a secure target design or implementation.

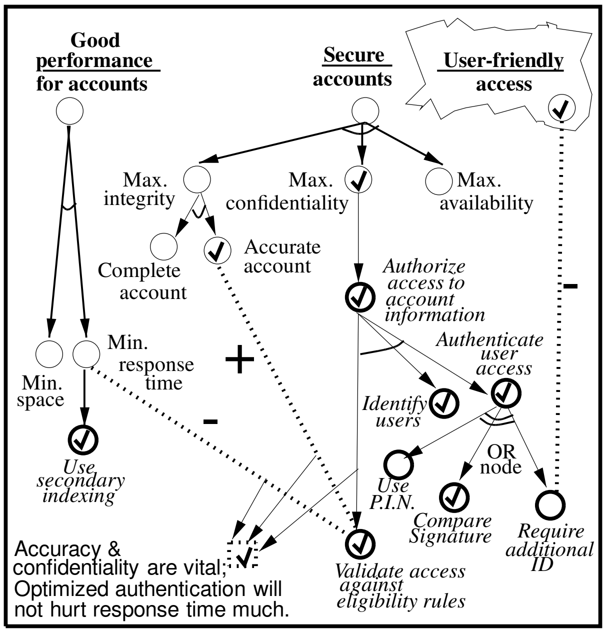
Assistant displays alternative techniques relevant to assuring Confidentiality, along with their relative trade-offs (shown in Figure 2).

Developer examines a catalogue of confidentiality assurance techniques displayed by the system, and decides to select an Authorization Technique.

Assistant creates and links techniques.

Repeating this process, the Authorization Technique is further refined to goals for Identification, Authentication, and Access Rule Validation.

3.3 Dealing with trade-offs.



—> Assistant automatically detected
Accuracy-Confidentiality synergy (+),
and warned against possible omission of
User-friendliness based on correlation rules.
Developer deals with Performance
—> Assistant detects Confidentiality-Time conflict (-).
Developer examines trade-offs
—> Assistant displays relevant arguments.
Developer makes and justifies decisions
—> Assistant evaluates goal satisfaction.

Figure 1(c). Dealing with trade-offs.

Assistant automatically detected earlier a synergy (+) between Accuracy and Confidentiality and created a link from *"validate access against eligibility rules"* to the Accurate account goal (Validation has a positive impact on the accuracy of accounts, as ill-intentioned users can be denied access and prevented from committing forgery.), when the **developer** decomposed the authorization technique into three sub-goals. This link is omitted from Figure 1(b) for the purposes of presentation.

Assistant detected the conflict (–) between *"Require additional ID"* and User-friendly access, when the **developer** selected alternative techniques for further refining *"Authenticate use access."* Shown here, and omitted from Figure 1(b), is the warning given by the **Assistant** that the developer has not considered the requirement on User-Friendly Access.

Developer deals with Performance Requirements, introducing the Minimum Response Time Requirement.

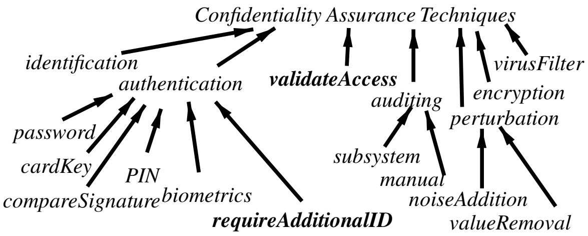
Assistant detects the conflict between the Minimum Response Time and Confidentiality Requirements. Validation induces extra overhead. It then records this relationship between *"validate access against eligibility rules"* and the Minimum Response Time goal.

Developer examines the goal synergy and conflict induced by the *"validate access against eligibility rules"*, and wants to find ways to resolve this situation.

Assistant displays relevant ways to make arguments in dealing with trade-off decisions.

Developer selects the *vital-few-trivial-many* method to support the decision to validate access against eligibility rules, and justifies the decision with argument. In effect, the developer invalidates the negative impact of the validation towards the response time.

Assistant evaluates goal satisfaction.



Technique / NFRGoal	Validate-Access rules	Authentication-requiringAdditionalID
Accuracy	+	
Confidentiality	+	+
UserFriendliness		-
Response Time	-	

Figure 2. A Portion of the Technique Hierarchy and Examples of Trade-offs.

The catalogue of alternative refinement methods and techniques, along with their trade-offs, are based on work done by researchers and practitioners in the particular areas such as security [ITSEC91, Parker91, Clark87, Martin73], and performance [Smith90, Hyslop91] of implementation of information systems [Nixon89].

Throughout the development process, both selected and discarded alternatives form part of the development history, and the Assistant keeps track of the impact of decisions upon the top-level goals.

4 Status

The NFR-Framework has been developed and described in a number of publications. The power of the framework has been illustrated using the following types of non-functional requirements, and applied to:

- *accuracy* [Chung91a,93a],
- *security* [Chung93a,b],
- *performance* [Nixon91,93,94],
- *user-friendliness* and *cost* [Chung93a].

The NFR-Assistant, a research prototype implementation, has been developed, including treatments for:

- *accuracy* [Chung93a],
- *security* [Chung93a,b], and
- (in progress) *performance* [Nixon94].

Studies have been conducted on a variety of information systems, including *credit card health insurance*, and *government administration* systems [Chung93a,b,c, Nixon93,94]. As our aim is to apply A.I. technology to real problems, our studies have used documents obtained from the organisations, including system descriptions, policy and procedure manuals, and workload statistics [Visa International91, Ontario80, Revenue Canada92]. In this way, our studies have addressed a variety of NFRs, a number of application areas (ranging from commercial to governmental), and systems with a variety of characteristics (ranging from a high volume of short-term transactions on a large information base, to a smaller volume of long-term processes). However, we have not yet worked closely with development teams from the organisations.

Evaluation and Limitations. We found that the NFR-Framework enabled us to *represent* the relevant concepts and methods for dealing with NFRs during the software development process. This was successful because methods offer a body of NFR-related *vocabulary* and *subject matter,* allowing us to succinctly capture a large number of NFR-specific concepts, such as security and performance, and their associated techniques, in an organised manner. We also found that the NFR-Framework enabled us to successfully *use* the above representations to relate NFRs to design decisions. While the coverage (hit ratio) of our methods was high for the studies already undertaken, the definition and use of more specialised methods would require additional expertise.

We were also able to use the framework to successfully *detect defects.* In some cases, this was made more straightforward, by using syntactic checking. In addition, the structuring and definitions of goals were used to detect omissions. In the studies, the system also provided support for detecting and dealing with conflicts and redundancies, while ambiguities were detected and reduced by clarifying the specification of individual goals.

Concerning current limitations, larger case studies will help determine if this framework can reduce rework and scrap, inducing shorter production time and lower cost. Scalability is one outstanding issue for the tool. We need to see if larger bodies of goals, methods and tradeoffs can be accommodated and graphically represented.

5 Related Work

Quality characteristics of software has been an important theme in software engineering for a long time [Boehm78]. Quite appropriately, various problems have been noted in the past by practitioners and researchers alike. The difficulty in dealing with requirements is convincingly reported in [Lindstrom93]. Software developers encounter significant instances of missing, incorrect, or inconsistent requirements details. These defects can lead to project failure, as can improper management of requirements, and the inability to trace requirements into components of design and testing. The importance of detecting these defects *early* has been emphasized in [Boehm87], since correction of design or implementation errors can be 100 times more costly than correction at the requirements phase.

The need for a systematic framework is further motivated in [Benzel89]. In practice, non-functional requirements are often retrofitted late in the development process or pursued in parallel but separately from functional design. These practices tend to result in systems which cannot be accredited, are more costly and less trustworthy.

Quantitative- and product-oriented approaches for addressing NFRs have been proposed (e.g., [Keller90]). These approaches have been evaluated with an emphasis on defect defection and reduction. For instance, [Linger93] provides an elegant evaluation of the cleanroom approach, whose emphasis is on carrying out different inspection tasks by independent teams. Similarly, [Schneider92] gives some 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 detecting defects *ex post facto* by way of inspecting general requirements. Also supporting a quantitative approach to software quality, Basili and Musa [Basili91] advocate models and metrics of the software engineering process from a management perspective. Unlike these, our qualitative, process-oriented approach focuses on NFRs using a semi-formal and systematic approach to using NFRs to drive the process of *generating* quality software.

Quality Function Deployment (QFD) (or *The House of Quality*) [Hauser88], one of the most advanced quality-related works in industrial engineering, has been applied to Software Quality Assurance and Improvement (e.g., [Zultner92, Yoshizawa90]). 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 benefits of design rationale.

Besides our own studies, in using the NFR-Framework for several software systems, there are other research experiences. The NFR-Framework has been adapted to modelling organizations during the development of information systems by one of the authors [Yu 93a,b,94, forthcoming]. It has also been adapted to address project risk management [Parmakson93]. The framework has been adapted for use in a large requirements engineering project [Jarke93], and is also a subject of a comparative study of goal-oriented approaches [Finkelstein93].

However, it awaits to be seen if the NFR-Framework is as effective in software quality engineering as QFD in industrial quality engineering [Sullivan86] [Kogure83], to reduce rework and scrap, hence inducing shorter production time and lower cost.

6 Conclusions: Application of the Framework

We view our work as a good start on the problem of addressing quality requirements systematically during the software development process. We would like to see the NFR-Framework and NFR-Assistant used by practising software engineers, and obtain their feedback.

We anticipate fruitful use of the framework by a variety of users. They may be dealing with a variety of *non-functional requirements* (not limited to accuracy, security, performance, etc.), a variety of *domains* (not limited to commercial, governmental and multi-sectoral), and a variety of *system characteristics* (including systems with large and small workload volumes, and short- and long-term processes).

Acknowledgements. Many thanks to Prof. John Mylopoulos for ongoing direction for this work. Our gratitude to Tracy Keeter for suggestions to improve the quality of the visuals.

Bibliography

[AI84] *Artificial Intelligence*, An International Journal, Special Volume on Qualitative Reasoning about Physical Systems, vol. 24, Nos. 1–3, Dec. 1984.

[Basili91] V. R. Basili and J. D. Musa, "The Future Engineering of Software: A Management Perspective," *IEEE Computer,* vol 24, no. 9, Sept. 1991, pp. 90–96.

[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.

[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.

[Chung93c] Lawrence Chung and Brian A. Nixon, Dealing with Non-Functional Requirements: Three Case Studies. Working paper, September 1993.

[Chung94] 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.

[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.

[Dardenne93] A. Dardenne, A. van Lamsweerde, S. Fickas, Goal-directed Requirements Acquisition. Science of Computer Programming Vol. 20, 1993, pp. 3–50.

[Fickas91] Stephen Fickas, Rob Helm, Martin Feather, When Things Go Wrong: Predicting Failure in Multi-Agent Systems. In Robert Balzer, John Mylopoulos (Workshop Co-Chairs), *International Workshop on the Development of Intelligent Information Systems,* Niagara-on-the-Lake, Ontario, April 21–23, 1991, pp. 47–53.

[Finkelstein93] Anthony C. W. Finkelstein and Stewart J. M. Green, *Goal-oriented Requirements Engineering.* Technical Report TR–93–42, Imperial College (London Univ.), forthcoming, 1993.

[Greenspan84] S. J. Greenspan, J. Mylopoulos and A. Borgida, "Capturing More World Knowledge in the Requirements Specification." *Proc., Sixth International Conference on Software Engineering,* 1982, pp. 225–234.

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

[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] M. Jarke, J. Bubenko, C. Rolland, A. Sutcliffe and Y. Vassiliou, "Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis," *Proc. of the IEEE Int. Symp. on Requirements Eng.,* San Diego, CA, January 4–6, 1993. Los Alamitos, CA: IEEE Computer Society Press, pp. 19–31.

[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.

[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.

[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.

[Lindstrom93] David R. Lindstrom, "Five Ways to Destroy a Development Project." *IEEE Software,* September 1993, pp. 55–58.

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

[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.

[Nixon89] Brian A. Nixon, K. Lawrence Chung, David Lauzon, Alex Borgida, John Mylopoulos and Martin Stanley, Design of a Compiler for a Semantic Data Model. In Joachim W. Schmidt and Costantino Thanos (Editors), *Foundations of Knowledge Base Management*. Berlin: Springer-Verlag, 1989, pp. 293–343.

[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] Ontario, "Chapter 17: The Ontario Health Insurance Plan Computer System," In *Report of the Commission of Inquiry into the Confidentiality of Health Information, Volume II*, 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.

[Ramesh92] Balasubramaniam Ramesh and Vasant Dhar, Supporting Systems Development by Capturing Deliberations During Requirements Engineering. *IEEE Transactions on Software Engineering,* Vol. 18, No. 6, June 1992, pp. 498–510.

[Revenue Canada92] 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.

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

[Yoshizawa90] T. Yoshizawa, "Quality Function Deployment for Software Development." *Second International Workshop on Software Quality Improvement,* Kyoto, Japan, January, 1990.

[Yu93a] 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.

[Yu93b] Eric S. K. Yu and John Mylopoulos, An Actor Dependency Model of Organizational Work — With Application to Business Process Reengineering. Proceedings, Conference on Organizational Computing Systems, 1993.

[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, "An Organization Modelling Framework for Information Systems Requirements Engineering", 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.

Author Biographies

Lawrence Chung received the B.Sc., M.Sc. and Ph.D. degrees in Computer Science from the University of Toronto in 1981, 1984 and 1993, respectively. He is currently an Assistant Professor of Computer Science at the University of Texas at Dallas. Previously he was a lecturer of several Computer Science courses at the University of Toronto. He has participated in the implementation of Taxis, an early object-oriented design language. His interests are in applications of artificial intelligence to software engineering, databases and languages. With experience in semantic data model implementation and information systems engineering, his current interests are in the representation and use of quality requirements in the software process.

Brian A. Nixon received the B.Sc. degree in Computer Science with a minor in Commerce in 1980, and the M.Sc. in Computer Science in 1983, both from the University of Toronto. He is now completing his Ph.D. studies in Computer Science at the University of Toronto. He has participated in the Taxis implementation project. His interests are in applications of artificial intelligence to databases, languages, and software engineering. His focus is on information system development, including implementation of semantic data models, and their performance.

Eric Yu received the B.A.Sc. degree in electrical engineering from the University of Toronto in 1974, and the M.Math. degree in Computer Science from the University of Waterloo in 1982. He was a Member of Scientific Staff at BNR and is now completing his Ph.D. studies in Computer Science at the University of Toronto. In his Ph.D. research, he is developing a framework for modelling organizations, and has applied it to information systems requirements engineering, organizational impact analysis, business process reengineering, and software process modelling. One component of the framework is based on the NFR framework described in this paper.