Lecture Notes in Computer Science 1943

Floor Koornneef
Meine van der Meulen (Eds.)

Computer Safety, Reliability and Security

19th International Conference, SAFECOMP 2000
Rotterdam, The Netherlands, October 2000
Proceedings

Springer

Floor Koornneef Meine van der Meulen (Eds.)

Computer Safety, Reliability and Security

19th International Conference, SAFECOMP 2000
Rotterdam, The Netherlands, October 24-27, 2000
Proceedings

Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Floor Koornneef
TU Delft, Safety Science Group
Jaffalaan 5, 2628 BX Delft, The Netherlands
E-mail: f.koornneef@tbm.tudelft.nl

Meine van der Meulen
SIMTECH
Max Euwelaan 60, 3062 MA Rotterdam, The Netherlands
E-mail: m.van.der.meulen@simtech.nl

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Computer safety, reliability and security : 19th international
conference ; proceedings / SAFECOMP 2000, Rotterdam, The Netherlands,
October 24 - 27, 2000. Floor Koornneef ; Meine van der Meulen (ed.). -
Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ;
Milan ; Paris ; Singapore ; Tokyo : Springer, 2000
(Lecture notes in computer science ; Vol. 1943)
ISBN 3-540-41186-0

CR Subject Classification (1998): D.1-4, E.4, C.3, F.3, K.6.5

ISSN 0302-9743
ISBN 3-540-41186-0 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

Printed in Germany

Typesetting: Camera-ready by author, data conversion by DA-TeX Gerd Blumenstein
Printed on acid-free paper SPIN: 10780880 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 1943
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Preface

Welcome to Rotterdam and to the International Conference Safecomp 2000, on the reliability, safety and security of critical computer applications. This already marks the 19th year of the conference, showing the undiminished interest the topic elicits from both academia and industry. Safecomp has proven to be an excellent place to meet and have discussions, and we hope this trend continues this year.

People and organisations depend more and more on the functioning of computers. Whether in household equipment, telecommunication systems, office applications, banking, people movers, process control or medical systems, the often-embedded computer subsystems are meant to let the hosting system realise its intended functions. The assurance of proper functioning of computers in dependable applications is far from obvious. The millennium started with the bug and the full endorsement of the framework standard IEC 61508. The variety of dependable computer applications increases daily, and so does the variety of risks related to these applications. The assessment of these risks therefore needs reflection and possibly new approaches. This year's Safecomp provides a broad mix of papers on these issues, on progress made in different application domains and on emerging challenges.

One of the special topics this year is *transport and infrastructure*. One would be hard pressed to find a better place to discuss this than in Rotterdam. The reliability, safety and security of computers is of prominent importance to Rotterdam, as a few examples illustrate. Its harbour depends on the reliable functioning of container handling systems, on the safe functioning of its radar systems, and, as of recently, on the safe and reliable functioning of the enormous storm surge barrier at Hoek van Holland.

A new topic for Safecomp is *medical systems*. These progressively depend on – embedded – programmable electronic systems. Experience shows that the medical world lacks the methods for applying these systems safely and reliably. We welcome a group of people ready to discuss this topic, and hope, by doing so, to contribute to this field of applications of safe, reliable and secure systems.

Software process improvement also represents a special topic of Safecomp 2000. It proved to be the most fruitful of the three in terms of submitted papers. There were many contributions from a host of countries, which had to be spread amongst different session topics.

We wish to thank the International Program Committee's members, 41 in total, for their efforts in reviewing the papers and for their valuable advice in organising this conference. We are also grateful for their contribution to distributing calls for papers and announcements. Without their help the burden of organising this conference would have been much greater.

Finally, let us once again welcome you to Rotterdam, a truly international city and home to people of many nationalities. We hope you take the time not only to enjoy this conference, but also to find your way around the city, since it surely has much to offer.

Floor Koornneef
Meine van der Meulen

Lecture Notes in Computer Science

For information about Vols. 1–1872
please contact your bookseller or Springer-Verlag

Vol. 1873: M. Ibrahim, J. Küng, N. Revell (Eds.), Database and Expert Systems Applications. Proceedings, 2000. XIX, 1005 pages. 2000.

Vol. 1874: Y. Kambayashi, M. Mohania, A M. Tjoa (Eds.), Data Warehousing and Knowledge Discovery. Proceedings, 2000. XII, 438 pages. 2000.

Vol. 1875: K. Bauknecht, S.K. Madria, G. Pernul (Eds.), Electronic Commerce and Web Technologies. Proceedings, 2000. XII, 488 pages. 2000.

Vol. 1876: F. J. Ferri, J.M. Iñesta, A. Amin, P. Pudil (Eds.), Advances in Pattern Recognition. Proceedings, 2000. XVIII, 901 pages. 2000.

Vol. 1877: C. Palamidessi (Ed.), CONCUR 2000 – Concurrency Theory. Proceedings, 2000. XI, 612 pages. 2000.

Vol. 1878: J.P. Bowen, S. Dunne, A. Galloway, S. King (Eds.), ZB 2000: Formal Specification and Development in Z and B. Proceedings, 2000. XIV, 511 pages. 2000.

Vol. 1879: M. Paterson (Ed.), Algorithms – ESA 2000. Proceedings, 2000. IX, 450 pages. 2000.

Vol. 1880: M. Bellare (Ed.), Advances in Cryptology – CRYPTO 2000. Proceedings, 2000. XI, 545 pages. 2000.

Vol. 1881: C. Zhang, V.-W. Soo (Eds.), Design and Applications of Intelligent Agents. Proceedings, 2000. X, 183 pages. 2000. (Subseries LNAI).

Vol. 1882: D. Kotz, F. Mattern (Eds.), Agent Systems, Mobile Agents, and Applications. Proceedings, 2000. XII, 275 pages. 2000.

Vol. 1883: B. Triggs, A. Zisserman, R. Szeliski (Eds.), Vision Algorithms: Theory and Practice. Proceedings, 1999. X, 383 pages. 2000.

Vol. 1884: J. Štuller, J. Pokorný, B. Thalheim, Y. Masunaga (Eds.), Current Issues in Databases and Information Systems. Proceedings, 2000. XIII, 396 pages. 2000.

Vol. 1885: K. Havelund, J. Penix, W. Visser (Eds.), SPIN Model Checking and Software Verification. Proceedings, 2000. X, 343 pages. 2000.

Vol. 1886: R. Mizoguchi, J. Slaney /Eds.), PRICAI 2000: Topics in Artificial Intelligence. Proceedings, 2000. XX, 835 pages. 2000. (Subseries LNAI).

Vol. 1888: G. Sommer, Y.Y. Zeevi (Eds.), Algebraic Frames for the Perception-Action Cycle. Proceedings, 2000. X, 349 pages. 2000.

Vol. 1889: M. Anderson, P. Cheng, V. Haarslev (Eds.), Theory and Application of Diagrams. Proceedings, 2000. XII, 504 pages. 2000. (Subseries LNAI).

Vol. 1890: C Linnhoff-Popien, H.-G. Hegering (Eds.), Trends in Distributed Systems: Towards a Universal Service Market. Proceedings, 2000. XI, 341 pages. 2000.

Vol. 1891: A.L. Oliveira (Ed.), Grammatical Inference: Algorithms and Applications. Proceedings, 2000. VIII, 313 pages. 2000. (Subseries LNAI).

Vol. 1892: P. Brusilovsky, O. Stock, C. Strapparava (Eds.), Adaptive Hypermedia and Adaptive Web-Based Systems. Proceedings, 2000. XIII, 422 pages. 2000.

Vol. 1893: M. Nielsen, B. Rovan (Eds.), Mathematical Foundations of Computer Science 2000. Proceedings, 2000. XIII, 710 pages. 2000.

Vol. 1894: R. Dechter (Ed.), Principles and Practice of Constraint Programming – CP 2000. Proceedings, 2000. XII, 556 pages. 2000.

Vol. 1895: F. Cuppens, Y. Deswarte, D. Gollmann, M. Waidner (Eds.), Computer Security – ESORICS 2000. Proceedings, 2000. X, 325 pages. 2000.

Vol. 1896: R. W. Hartenstein, H. Grünbacher (Eds.), Field-Programmable Logic and Applications. Proceedings, 2000. XVII, 856 pages. 2000.

Vol. 1897: J. Gutknecht, W. Weck (Eds.), Modular Programming Languages. Proceedings, 2000. XII, 299 pages. 2000.

Vol. 1898: E. Blanzieri, L. Portinale (Eds.), Advances in Case-Based Reasoning. Proceedings, 2000. XII, 530 pages. 2000. (Subseries LNAI).

Vol. 1899: H.-H. Nagel, F.J. Perales López (Eds.), Articulated Motion and Deformable Objects. Proceedings, 2000. X, 183 pages. 2000.

Vol. 1900: A. Bode, T. Ludwig, W. Karl, R. Wismüller (Eds.), Euro-Par 2000 Parallel Processing. Proceedings, 2000. XXXV, 1368 pages. 2000.

Vol. 1901: O. Etzion, P. Scheuermann (Eds.), Cooperative Information Systems. Proceedings, 2000. XI, 336 pages. 2000.

Vol. 1902: P. Sojka, I. Kopeček, K. Pala (Eds.), Text, Speech and Dialogue. Proceedings, 2000. XIII, 463 pages. 2000. (Subseries LNAI).

Vol. 1903: S. Reich, K.M. Anderson (Eds.), Open Hypermedia Systems and Structural Computing. Proceedings, 2000. VIII, 187 pages. 2000.

Vol. 1904: S.A. Cerri, D. Dochev (Eds.), Artificial Intelligence: Methodology, Systems, and Applications. Proceedings, 2000. XII, 366 pages. 2000. (Subseries LNAI).

Vol. 1905: H. Scholten, M.J. van Sinderen (Eds.), Interactive Distributed Multimedia Systems and Telecommunication Services. Proceedings, 2000. XI, 306 pages. 2000.

Vol. 1906: A. Porto, G.-C. Roman (Eds.), Coordination Languages and Models. Proceedings, 2000. IX, 353 pages. 2000.

Vol. 1907: H. Debar, L. Mé, S.F. Wu (Eds.), Recent Advances in Intrusion Detection. Proceedings, 2000. X, 227 pages. 2000.

Vol. 1908: J. Dongarra, P. Kacsuk, N. Podhorszki (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface. Proceedings, 2000. XV, 364 pages. 2000.

Vol. 1909: T. Yakhno (Ed.), Advances in Information Systems. Proceedings, 2000. XVI, 460 pages. 2000.

Vol. 1910: D.A. Zighed, J. Komorowski, J. Żytkow (Eds.), Principles of Data Mining and Knowledge Discovery. Proceedings, 2000. XV, 701 pages. 2000. (Subseries LNAI).

Vol. 1911: D.G. Feitelson, L. Rudolph (Eds.), Job Scheduling Strategies for Parallel Processing. VII, 209 pages. 2000.

Vol. 1912: Y. Gurevich, P.W. Kutter, M. Odersky, L. Thiele (Eds.), Abstract State Machines. Proceedings, 2000. X, 381 pages. 2000.

Vol. 1913: K. Jansen, S. Khuller (Eds.), Approximation Algorithms for Combinatorial Optimization. Proceedings, 2000. IX, 275 pages. 2000.

Vol. 1914: M. Herlihy (Ed.), Distributed Computing. Proceedings, 2000. VIII, 389 pages. 2000.

Vol. 1915: S. Dwarkadas (Ed.), Languages, Compilers, and Run-Time Systems for Scalable Computers. Proceedings, 2000. VIII, 301 pages. 2000.

Vol. 1916: F. Dignum, M. Greaves (Eds.), Issues in Agent Communication. X, 351 pages. 2000. (Subseries LNAI).

Vol. 1917: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature – PPSN VI. Proceedings, 2000. XXI, 914 pages. 2000.

Vol. 1918: D. Soudris, P. Pirsch, E. Barke (Eds.), Integrated Circuit Design. Proceedings, 2000. XII, 338 pages. 2000.

Vol. 1919: M. Ojeda-Aciego, I.P. de Guzman, G. Brewka, L. Moniz Pereira (Eds.), Logics in Artificial Intelligence. Proceedings, 2000. XI, 407 pages. 2000. (Subseries LNAI).

Vol. 1920: A.H.F. Laender, S.W. Liddle, V.C. Storey (Eds.), Conceptual Modeling – ER 2000. Proceedings, 2000. XV, 588 pages. 2000.

Vol. 1921: S.W. Liddle, H.C. Mayr, B. Thalheim (Eds.), Conceptual Modeling for E-Business and the Web. Proceedings, 2000. X, 179 pages. 2000.

Vol. 1922: J. Crowcroft, J. Roberts, M.I. Smirnov (Eds.), Quality of Future Internet Services. Proceedings, 2000. XI, 368 pages. 2000.

Vol. 1923: J. Borbinha, T. Baker (Eds.), Research and Advanced Technology for Digital Libraries. Proceedings, 2000. XVII, 513 pages. 2000.

Vol. 1924: W. Taha (Ed.), Semantics, Applications, and Implementation of Program Generation. Proceedings, 2000. VIII, 231 pages. 2000.

Vol. 1925: J. Cussens, S. Džeroski (Eds.), Learning Language in Logic. X, 301 pages 2000. (Subseries LNAI).

Vol. 1926: M. Joseph (Ed.), Formal Techniques in Real-Time and Fault-Tolerant Systems. Proceedings, 2000. X, 305 pages. 2000.

Vol. 1927: P. Thomas, H.W. Gellersen, (Eds.), Handheld and Ubiquitous Computing. Proceedings, 2000. X, 249 pages. 2000.

Vol. 1928: U. Brandes, D. Wagner (Eds.), Graph-Theoretic Concepts in Computer Science. Proceedings, 2000. X, 315 pages. 2000.

Vol. 1929: R. Laurini (Ed.), Advances in Visual Information Systems. Proceedings, 2000. XII, 542 pages. 2000.

Vol. 1931: E. Horlait (Ed.), Mobile Agents for Telecommunication Applications. Proceedings, 2000. IX, 271 pages. 2000.

Vol. 1658: J. Baumann, Mobile Agents: Control Algorithms. XIX, 161 pages. 2000.

Vol. 1766: M. Jazayeri, R.G.K. Loos, D.R. Musser (Eds.), Generic Programming. Proceedings, 1998. X, 269 pages. 2000.

Vol. 1791: D. Fensel, Problem-Solving Methods. XII, 153 pages. 2000. (Subseries LNAI).

Vol. 1799: K. Czarnecki, U.W. Eisenecker, Generative and Component-Based Software Engineering. Proceedings, 1999. VIII, 225 pages. 2000.

Vol. 1812: J. Wyatt, J. Demiris (Eds.), Advances in Robot Learning. Proceedings, 1999. VII, 165 pages. 2000. (Subseries LNAI).

Vol. 1932: Z.W. Raś, S. Ohsuga (Eds.), Foundations of Intelligent Systems. Proceedings, 2000. XII, 646 pages. (Subseries LNAI).

Vol. 1933: R.W. Brause, E. Hanisch (Eds.), Medical Data Analysis. Proceedings, 2000. XI, 316 pages. 2000.

Vol. 1934: J.S. White (Ed.), Envisioning Machine Translation in the Information Future. Proceedings, 2000. XV, 254 pages. 2000. (Subseries LNAI).

Vol. 1935: S.L. Delp, A.M. DiGioia, B. Jaramaz (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2000. Proceedings, 2000. XXV, 1250 pages. 2000.

Vol. 1937: R. Dieng, O. Corby (Eds.), Knowledge Engineering and Knowledge Management. Proceedings, 2000. XIII, 457 pages. 2000. (Subseries LNAI).

Vol. 1938: S. Rao, K.I. Sletta (Eds.), Next Generation Networks. Proceedings, 2000. XI, 392 pages. 2000.

Vol. 1939: A. Evans, S. Kent, B. Selic (Eds.), «UML» – The Unified Modeling Language. Proceedings, 2000. XIV, 572 pages. 2000.

Vol. 1940: M. Valero, K. Joe, M. Kitsuregawa, H. Tanaka (Eds.), High Performance Computing. Proceedings, 2000. XV, 595 pages. 2000.

Vol. 1942: H. Yasuda (Ed.), Active Networks. Proceedings, 2000. XI, 424 pages. 2000.

Vol. 1943: F. Koornneef, M. van der Meulen (Eds.), Computer Safety, Reliability and Security. Proceedings, 2000. X, 432 pages. 2000.

Vol. 1945: W. Grieskamp, T. Santen, B. Stoddart (Eds.), Integrated Formal Methods. Proceedings, 2000. X, 441 pages. 2000.

Vol. 1948: T. Tan, Y. Shi, W. Gao (Eds.), Advances in Multimodal Interfaces – ICMI 2000. Proceedings. [illegible]0. XVI, 678 pages. 2000.

Vol. 1954: W.A. Hunt, Jr., S.D. Johnson (Eds.), Formal Methods in Computer-Aided Design. Proceedings, 2000. XI, 539 pages. 2000.

Table of Contents

Formal Methods

Invited Paper

Safety Guidelines, Standards and Certification

Hardware Aspects

Safety Assessment I

Design for Safety

Invited Paper

Transport & Infrastructure

Safety Assessment II

The Ten Most Powerful Principles for Quality in (Software and) Software Organizations for Dependable Systems

Tom Gilb

Result Planning Limited,
Iver Holtersvei 2, N-1410 Kolbotn, Norway
Phone: +(47) 66 80 46 88, Mobile: +(47) 926 67187
Gilb@acm.org
http://www.Result-Planning.com,

Abstract. Software knows it has a problem. Solutions abound, but which solutions work? What are the most fundamental underlying principles we can observe in successful projects? This paper presents 10 powerful principles that are not widely taught or appreciated. They are based on ideas of measurement, quantification and feedback. Our maturity level with respect to 'numbers' is known to be poor. Hopefully, as we move to higher maturity levels we will also begin to appreciate the power of measurement and numeric expression of idea. What can we do right now? I suggest the first step is to recognize that all your quality requirements can and should be specified numerically. I am not talking about 'counting bugs'. I am talking about quantifying qualities such as security, portability, adaptability, maintainability, robustness, usability, reliability and performance. Decide to make them numeric on your project. Draft some numeric requirements today, surprise your team tomorrow!

1 Introduction

All projects have some degree of failure, compared to initial plans and promises. Far too many software projects fail totally. In the mid 1990s, the US Department of Defense estimated that about half of their software projects were total failures! (Source: N Brown). The civil sector is no better [16]. So what can be done to improve project success? This paper outlines ten key principles of successful software development methods, which characterize best practice.

These 10 most powerful software quality principles are selected because there is practical experience showing that they really get us control over qualities, and over the costs of qualities. They have a real track record. This record often spans decades of practice in companies like IBM, HP and Raytheon. There is nothing 'new' about them. They are classic. But the majority of our community is young and experientially

F. Koornneef and M. van der Meulen (Eds.): SAFECOMP 2000, LNCS 1943, pp. 1-13, 2000.

new to the game, so my job is to remind us of the things that work well. Your job is to evaluate this information and start getting the improvements your management wants in terms of quality and the time and effort needed to get it.

"Those who do not learn from history, are doomed to repeat it" (Santayana, 1903, The Life of Reason).

Principle 1: Use Feedback

Experience of formal feedback methods is decades old, and many do appreciate their power. However, far too many software engineers and their managers are still practicing low-feedback methods, such as Waterfall project management (also known as Big Bang, Grand Design). Far too many also are checking the quality of their systems by relying on testing, 'late in the day', when they have finished producing their entire system. Even many textbooks and courses continue to present low-feedback methods. This is not from conscious rejection of high-feedback methods, but from ignorance of the many successful and well-documented projects, which have detailed the value of feedback.

Methods using feedback succeed; those without seem to fail. 'Feedback' is the single most powerful principle for software engineering. (Most of the other principles in this paper are really ideas, which support the use of feedback.) Feedback helps you get better control of your project by providing facts about how things are working in practice. Of course, the presumption is that the feedback is early enough to do some good. This is the crux: rapid feedback. We have to have the project time to make use of the feedback (for example, to radically change direction, if that is necessary). Some of the most notable rapid high-feedback methods include:

Defect Prevention Process (originated Mays and Jones, IBM 1983) The Defect Prevention Process (DPP) equates to Software Engineering Institute CMM Level 5 as practiced at IBM from 1983-1985 and on [14]. DPP is a successful way to remove the root causes of defects. In the short term (a year) about 50% defect reduction can be expected; within 2-3 years, 70% reduction (compared to original level) can be experienced and over a 5-8 year timeframe, 95% defect reduction is possible (Sources: IBM Experience, Raytheon Experience [5]).

The key feedback idea is to 'decentralize' the initial causal analysis activity investigating defects to the grass roots programmers and analysts. This gives you the true causes and acceptable, realistic change suggestions. Deeper 'cause analysis' and 'measured process-correction' work can then be undertaken outside of deadline-driven projects by the more specialized and centralized Process Improvement Teams.

The feedback mechanisms are many. For example, same-day feedback is obtained from the people working with the specification and, early numeric process change-result feedback is obtained from the Process Improvement Teams.

Inspection (originated Fagan, IBM 1975) The Inspection method originated in IBM in work carried out by M. Fagan, H. Mills ('Cleanroom') and R. Radice (CMM inventor). It was originally primarily focussed on bug removal in code and code design

documents. Many continue to use it in this way today. However, Inspection has changed character in recent years. Today, it can be used more cost-effectively by focussing on measuring the Major defect level (software standards violations) in sample areas (rather than processing the entire document) of any software or upstream marketing specifications [9]. The defect level measurement should be used to decide whether the entire specification is fit for release (exit) downstream to be used, say for a 'go/no-go' decision-making review or for further refinement (test planning, design, coding).
The main Inspection feedback components are:

- feedback to author from colleagues regarding compliance with software standards.
- feedback to author about required levels of standards compliance in order to consider their work releasable.

Evolutionary Project Management (originated within 'Cleanroom' methods, Mills, IBM 1970) Evolutionary Project Management (Evo) has been successfully used on the most demanding space and military projects since 1970 [15], [13], [2], [8], [10]. The US Department of Defense changed their software engineering standard (2167a) to an Evo standard (MIL-STD-498, which derived succeeding public standards (for example, IEEE)). The reports (op. cit.) and my own experience, is that Evo results in a remarkable ability to delivery on time and to budget, or better, compared to conventional project management methods [16].

An Evo project is consciously divided up into small, early and frequently delivered, stakeholder result-focussed steps. Each step delivers benefit and build towards satisfaction of final requirements. Step size is typically weekly or 2% of total time or budget. This results in excellent regular and realistic feedback about the team's ability to deliver meaningful measurable results to selected stakeholders. The feedback includes information on design suitability, stakeholders' reaction, requirements' tradeoffs, cost estimation, time estimation, people resource estimation, and development process aspects.

Statistical Process Control [originated Shewhart, Deming, Juran: from 1920's] Statistical Process Control (SPC) although widely used in manufacturing [4] is only to a limited degree actually used in software work. Some use is found in advanced Inspections [5],[18]. The Plan Do Study (or Check) Act cycle is widely appreciated as a fundamental feedback mechanism.

Principle 2: Identify Critical Measures

It is true of any system, that there are several factors, which can cause a system to die. It is true of your body, your organization, your project and your software or service product. Managers call them 'Critical Success Factors.' If you analyzed systems looking for all the critical factors, which caused shortfalls or failures, you would get a list of factors needing better control. They would include both stakeholder values (such as serviceability, reliability, adaptability, portability and usability) and the critical resources needed to deliver those values (such as people, time, money and data

quality). You would find, for each of these critical factors, a series of faults, which would include:

- failure to systematically identify all critical stakeholders and their critical needs
- failure to define the factor measurably. Typically, only buzzwords are used and no indication is given of the survival failure) and target (success) measures
- failure to define a practical way to measure the factor
- failure to contract measurably for the critical factor
- failure to design towards reaching the factor's critical levels
- failure to make the entire project team aware of the numeric levels needed for the critical factors
- failure to maintain critical levels of performance during peak loads or on system growth.

Our entire culture and literature of 'software requirements' systematically fails to account for the majority of critical factors. Usually, only a handful, such as performance, financial budget and deadline dates are specified. Most quality factors are not defined quantitatively at all. In practice, all critical measures should always be defined with a useful scale of measure. However, people are not trained to do this and managers are no exception. The result is that our ability to define critical 'breakdown' levels of performance and to manage successful delivery is destroyed from the outset.

Principle 3: Control Multiple Objectives

You do not have the luxury of managing qualities and costs at whim. You cannot decide for a software project to manage just a few of the critical factors, and avoid dealing with the others. You have to deal with *all* the potential threats to your project, organization or system. You must simultaneously track and manage all the critical factors. If not, then the 'forgotten factors' will probably be the very reasons for project or system failure.

I have developed the Impact Estimation (IE) method to enable tracking of critical factors, but it does rely on critical objectives and quantitative goals having been identified and specified. Given that most software engineers have not yet learned to specify *all* their critical factors *quantitatively* (Principle 2), this *next* step, tracking progress against quantitative goals (this principle), is usually impossible.

IE is conceptually similar to Quality Function Deployment [1], but it is much more objective and numeric. It gives a picture of reality that can be monitored [8], [10]. See Table 1, an example of an IE table. It is beyond the scope of this paper to provide all the underlying detail for IE. To give a brief outline, the percentage (%) estimates (see Table 1) are based, as far as possible, on source-quoted, credibility evaluated, objective documented evidence. IE can be used to evaluate ideas *before* their application, and it can also be used (as in Table 1) to track progress towards multiple objectives *during* an Evolutionary project. In Table 1, the 'Actual' and 'Difference" and 'Total' numbers represent *feedback* in small steps for the chosen set of critical factors that management has decided to monitor. If the project is deviating from plans, this will be easily visible and can be corrected on the next step.