Monograph: Experiences and Advances in Software Quality
(published jointly with Novática*)

Guest Editors: Darren Dalcher and Luis Fernández-Sanz

2 Presentation: Improving Quality in Business Processes, Products and Organizational Systems — Darren Dalcher and Luis Fernández-Sanz

6 Preventative Software Quality Control: Using Human Checking to Change Defective Human Practice — Tom Gilb and Lindsey Brodie

14 The Software Process Improvement Hype Cycle — Miklós Biró

21 Quality Going for Gold — Derek Irving and Margaret Ross

26 Can Teamwork Management Help in Software Quality and Process Improvement? — Esperança Amengual-Alcover and Antònia Mas-Picacho

34 Evidence-based Software Engineering and Systematic Literature Reviews — Barbara Kitchenham, David Budgen, and O. Pearl Breton

42 Software Project Success: Moving Beyond Failure — Darren Dalcher

51 Software Measurement for Better Project and Process Quality — Christof Ebert


67 A Quality Evaluation Model for Web2.0 e-Learning Systems — Stephanos Mavromoustakos and Katerina Papanikolaou

UPENET (UPGRADE European NETwork)

75 From Mondo Digitale (AICA, Italy)

History of Computing

The Turing Test: History and Significance — Giuseppe O. Longo

CEPIS NEWS

89 Selected CEPIS News — Fiona Fanning

* This monograph will be also published in Spanish (full version printed; summary, abstracts, and some articles online) by Novática, journal of the Spanish CEPIS society ATI (Asociación de Técnicos de Informática) at <http://www.ati.es/novatica/>.
Quality is intangible, yet it has been in the forefront of software development and software engineering for well over thirty years. During this period the perception of quality has shifted from an activity that takes place downstream, to a management constraint that defines the success of the ultimate product and, beyond that, to a set of quality standards and improvement approaches. Experience has taught us that quality cannot be engineered into the later stages of development and must instead be purposefully designed into the process, product and organizational systems. Consequently, many organizations search for new ways to improve their business processes, increase their maturity profile, and enhance the efficiency and effectiveness of their software development practices.

This special issue brings together a collection of reflections and experiences from leading experts in the field of software quality. Many of the papers report on new ideas and advances thereby offering novel perspectives and approaches for improving quality in software. The papers are grounded in both research and practice and therefore deliver insights that summarize the state of the discipline while indicating avenues for improvement and placing new trends in the context of improving quality in an organizational setting.

The nine papers selected for the issue showcase three perspectives in terms of the trends identified within the software quality domain. The first three papers report on new initiatives and the continuing evolution and improvement of older ideas (mainly in relation to process improvement). The next four papers introduce new ways of thinking and working in practice, while the final two papers feature new technologies and their impact on quality.

Inspections offer a well-established formal review mechanism originally developed for use on the source code of a program or a portion of a program as a systematic procedure for the detection of defects. The review is conducted by peers who attempt to look at the document from a wider perspective with the aim of uncovering lapses that may have
Eluded the author. While there are benefits associated with peer reviews, the process is tedious, demanding and extremely expensive. The paper by Gibb and Brodie makes a case for moving the inspection process upstream to uncover the error density earlier in the process and for sampling quality levels rather than spending time fixing the defects. Sampling facilitates more informed decision making, while emphasizing the need for quality standards and thus results in defect prevention and appreciation of the need to eliminate errors. Moreover, the results of defect analysis, which offer a short term improvement, also reveal where and how defects occur and can therefore be used as the basis for process improvement. Preventive software inspections as introduced in this paper represent a departure from accepted practices with a real potential to improve quality processes and culture within organizations.

One of the more practical streams within software quality practice has concentrated on the idea of software process improvement (SPI), and has spawned numerous maturity models and capability frameworks. SPI advocates the systematic improvement of software processes by assessing software processes against process standards and frameworks and by mapping levels of achievement. The paper by Biró surveys the historical development of process improvement and its impact on the software practice using the hype cycle as a lens. The adoption of CMM by the US Department of Defense has accelerated the rate of adoption of SPI, giving additional legitimacy to this burgeoning area. The paper maps additional developments such as ISO 9000, Bootstrap, ISO/IEC 15504, explaining their role in the cycle associated with the SPI movement. The ideas of maturity and capability have been exported to other domains and disciplines but have also been criticized for their bureaucracy and inflexibility. Additional perspectives such as the one offered by agile methods may provide a new ground for the coming together of SPI methods and approaches.

The most common SPI approaches (CMMi and ISO 15504) have been implemented in a wide variety of environments and sectors. Consequently, a considerable number of companies and institutions have amassed practical experience of their strengths and weaknesses. Although the common methods have evolved over time, they seem to have become more stable. Other SPI approaches continue to evolve to address existing and future challenges. This indeed is the case of one of the traditional references in the area of software quality: the United Kingdom’s TickIT, a quality system combined with a certification scheme. Irving and Ross present the new scheme for TickITPlus to be launched in 2010. The new framework aims to cover various needs of organizations and industry. The paper “Quality: Going for Gold” is a good example of how an article can offer readers useful information regarding a forthcoming trend.

However tempting the idea may be, we simply cannot rely on new methods and processes to solve all our problems. A key concern in quality, whether in the software development domain or more generally, has always revolved around the human factor. Moreover, software development efforts tend to be human resource intensive. Although people are clearly central to development, not much rigorous and practical research has been carried out to explore the implications for software quality. “Can Teamwork Management Help in Software Quality and Process Improvement?” represents one clear contribution to exploring the topic. The work by Mas and Amengual addresses one of the most critical items for software development: effective and efficient teamwork. The relationship between teamwork and SPI methods is also analysed in the paper.

Evidence-based practice traces its roots to evidence-based medicine and its concern with a process of systematically finding, appraising and utilizing findings as the basis for clinical decisions. It has been adopted in many domains and disciplines primarily due to its success in unlocking vast data resources which inform decisions and provide a wider and better informed basis for identifying effective remedies for individual cases. Kitchenham, Budgen and Brereton have been instrumental in importing the idea of providing evidence from research integrated with practical experience and human values to improve decision making to the field of software engineering. Systematic literature reviews aggregate empirical results in a methodical way. They have the power to overturn “common knowledge” and to uncover additional evidence which may point to alternative explanations and practices. Many decisions made in software engineering suffer from insufficient data. The approaches proposed in this paper encourage practitioners to consider the scientific evidence underpinning their decisions and can pave the way towards improving the basis for making decisions. In the longer term it may lead to the development of a body of empirical evidence that can be used to improve decision making in software practice.

Various studies suggest that a large proportion of software development projects fail either fully or partially. The normal measure in such studies is the ability to meet initial targets for cost, time and performance. The paper by Dalcher argues for a move beyond simplistic failure studies which are predicated on internal project management efficiency criteria. Indeed, the fascination with failure needs to be replaced with a healthier interest in what is required to build success. In practice, success extends beyond such simplistic internal measures. Many of the issues identified in following the analysis of failure and success stories are to do with relationship management, politics, trust, expectations and escalation; factors that are not included in the typical failure surveys. A wider perspective with multiple representations of the levels of success is developed which expands the time horizons and looks at effectiveness, rather than efficiency, and the outcomes of a project. Effectiveness re-establishes quality as a primary consideration in determining the success of a project (thereby supplanting the delivery to pre-defined budget and schedule as the ultimate criteria). Success is complex and multi-layered concept. By re-framing our focus of interest from failure to success we may finally be able to progress the discussion.
about success track records in delivering software.

As we have seen software development can be confusing, sending us mixed messages. Given this messiness, how do we know what we need to measure? The failure or success of projects, achievement of quality objectives... How can one know what is happening in each case? Deming’s statement “In God we trust, all others bring data” is a good expression of we mere mortals’ need for measurement in trying to make sense of the environment. Traditionally, the software development community has been reluctant to compile comprehensive and systematic collections of data. Ebbert, the author of a recent book, offers a good review of the discipline, covering foundations as well as projects and process measurement. One of the strengths of this paper is the linking of metrics to SPI and project performance with practical examples.

Although, as shown in the previous papers, the software quality discipline embraces a considerable variety of approaches, methods and techniques, the field is also highly dependent on the evolution of software engineering. As we have seen throughout the emergence of earlier development paradigms (e.g. object orientation, UML, web engineering, etc.), software quality needs to engage, respond and react when new scenarios arise. One of the recent trends in software engineering is the service orientation of software. Following its emergence into mainstream, SOA (Service Oriented Architecture) is pushing the advancement of quality methods for software engineering. The team from the Universidad de Oviedo (García-Fanjul, Palacios, Tuya, and De la Riva) analyses the challenges of testing software services compositions overcoming limitations of traditional testing approaches with practical consequences for BPEL (de facto SOA standard in industry).

But it is not only specific advances in software engineering methods or approaches that drive the evolution of the software quality discipline. New types of software and systems also require adaptation and evolution of traditional methods (or even the creation of new ones) to address their specific characteristics. Mavromoustakos and Papanikolaou have accepted the challenge of contributing to software quality evaluation of systems with a widespread presence in our daily life: E-learning systems with web 2.0 capabilities. Their detailed work provides a comprehensive review of the many aspects that need to be taken into account for an evaluation with the support of a commonly accepted standard such as ISO 9126.

The initial interest in software quality was triggered by concerns about the performance of software and the perception of poor success rates associated with software development projects. Many advances have been proposed over the years and the practice has improved and become better established. As we have seen software development remains a challenging occupation which forces software quality experts to invent new methods and approaches, develop new perspectives and respond to new technologies. While the silver bullet is still missing, it is interesting to observe the vitality and diversity that continue to define and revitalize the area. In this issue, leading researchers and practitioners have surveyed the development of ideas, perspectives and concepts within software quality and given us a glimpse of some of the potential solutions. There is still a long way to go but the journey now seems more exciting.
Useful References about Software Quality

The following references, along with those included in the articles this monograph consists of, will help our readers to dig deeper into this field.

Books


Articles and Papers


Journal Special Issues Dedicated to Quality Topics

- Software Testing, a special issue of IEEE Software containing a number of papers on testing practices, and agile software testing, IEEE Software23(4), July/August 2006.
- Quality Requirements, a special issue of IEEE Software containing a number of articles on quality requirements as the front end of attaining quality, IEEE Software, 25(2), March/April 2008.

Web Sites