Food for Thought™

An e-newsletter published by
Software Quality Consulting, Inc.

November 2009 , Vol. 6 No. 6
[Text-only Version]

Welcome to ***Food for Thought***™, an e-newsletter from **Software Quality Consulting**. I've created free subscriptions for my valued business contacts. If you find this newsletter informative, I encourage you to continue reading. Feel free to pass this newsletter along to colleagues by clicking on the **Forward Email** link at the bottom of this email. If you've received this newsletter from a colleague and would like to subscribe, please click this **Enter New Subscription** link. If you don't wish to receive this newsletter, click the **SafeUnSubscribe**™ link at the bottom of this newsletter, and you won't be bothered again.

Your continued feedback on this newsletter is most welcome. Please send your comments and suggestions to **info@swqual.com**.

In this issue

In **This Months' Topic**, I conclude my discussion on the state of the software quality assurance profession...

Regular features to look for each month are:

- **Monthly Morsels**
  Hints, tips, techniques and reference info related to this month's topic
- **Calendar**
  Conferences, workshops, and meetings of interest to software engineers, QA engineers and anyone interested in software development

This month's topic

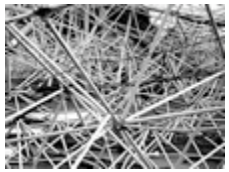
Software Quality Assurance turns 50
A critical look at the state of the profession

Part 3 - The Future of SQA

Software Quality Assurance (SQA), as we know it, was first applied to software development projects about 50 years ago. To recognize this important milestone, the state of the SQA profession has been the topic of my e-newsletter for the past few issues. In Part I of this series, I discussed the history and evolution of SQA. Part II focused on the current state of the profession - including successes and failures. This last installment is about the future of SQA.

A View of Software Development in the Future

Before we can discuss the future of SQA, we need some context for what software development will likely be like in the not-too-distant future. There have been several trends that will likely continue over the next decade. These include:



Software complexity is increasing at an exponential rate.

- **Increasing Complexity**

  As we have already seen, complexity is rising exponentially. Long gone are the days where software engineers understood all aspects of an application. In the future, software engineers will likely understand less and less of the overall software design. This will certainly affect quality, testability, and user satisfaction.

- **Geographically Dispersed Development Teams**

  Geographically dispersed development teams became commonplace in the 1990s with the advent of off shore development and testing. This trend has continued to evolve, even though several off shore projects have

failed. Off shore development and testing will continue expand in third world countries with poor infrastructure and low wages. Language and cultural problems will continue to pose significant challenges as will adequate cyber-security protection. The rise and fall of international currencies against the dollar will also continue to dictate the economic feasibility of this whole endeavor.

The increased use of geographically dispersed development and testing teams will create significant knowledge gaps as more and more critical design information falls through the proverbial cracks.

Read more about the future of off-shoring

- **Continuous Integration**

  Continuous integration started in the ‘90s and initially was a mostly manual process. Continuous integration processes and tools are evolving and in the future, they will become part of the normal day-to-day part of most software development projects.

  Read more about a typical open source Continuous Integration Tool and an extensible continuous integration server

- **Continuous Testing**

  For many years, software engineers were less than enthusiastic about unit testing their code. While techniques like the buddy system (you test mine, I’ll test yours) has led to some improvement, the real improvement will come from tools that are better at automating the unit testing process so that it can happen automatically and continuously - with every change made to the code. Such tools are becoming available and I expect them to be widely used over the next decade...

Read more about continuous testing tools for Ruby and Eclipse

- **Distributed Source Code Control**

  Source code control has always been a potential source of problems especially when teams are geographically dispersed. To be successful in the next decade, these teams will rely on a new breed of distributed source code tools that prevent many common problems and help manage the constantly changing code base.

  Read more about Git - a distributed source control tool

- **Continuous Deployment**

  In the past, many software development organizations agonized over how often should new releases be made available to customers. At one extreme are traditional organizations that release new versions very infrequently (yearly or quarterly). At the other extreme are organizations releasing new versions daily. Clearly, the deployment strategy depends on many factors, not the least of which is the ability to develop and test quickly. The trend will definitely be moving towards releasing more often.

  Read more about one company's continuous deployment model

What will SQA look like in 10 years?

So what does the future of SQA look like?

With this view of software development providing some context, we can now look at the future of SQA for the next decade or so...

- **Virtualization**

  Setting up a test lab has always been a challenge - both

technically and financially. Test groups never have enough systems to perform adequate testing in a timely manner. And maintaining test lab configurations is very time consuming.

In the next decade, we will see the increased use of virtual test labs - where different virtual platforms can be easily set up and configured to more closely match expected customer environments.

Read more about virtualization of testing labs

- **Test Automation Toolsand Frameworks**

  Test automation tools will continue to evolve and improve in the next decade. Newer test automation tools will require less programming skills and will include features to support:

  - Keyword-driven testing
  - Model-driven testing
  - GUI-driven testing

  The holy grail for test automation tools has always been a comprehensive test framework that supports the entire test development life cycle - everything from initial test planning, test prototyping, test development, test execution, defect reporting and tracking, and test reporting, metrics, and trending.

  There are some encouraging signs that test automation tool suppliers are finally starting to move in this direction. In addition, there will likely also be tighter integration between test automation tools - typically used by SQA engineers and continuous test tools - typically used by developers.

- **Balanced Test Teams**

Future test teams will likely be more balanced and will include new testing roles such as:

- **Test Architect** - analogous to a software architect, a test architect determines what kinds of tests are needed, what test tools would be most appropriate, and what test team skills are required. Test architects will typically have training in software engineering and testing.
- **Test Developer** - analogous to a software engineer, the test developer would be skilled in the scripting languages of several test automation tools. Test developers would create automated tests by collaborating with test architects and test engineers. Test developers will typically have training in software engineering and testing.
- **Test Engineer** - has domain knowledge and testing skills to create manual tests. Works with test architects and test developers to ensure that domain knowledge is applied in all of the many kinds of tests that are developed. Creates and executes manual and automated tests. Prepares test reports and statistics. Test engineers will typically have training in effective testing techniques.

- **Safety Cases**

  The recent **National Research Council** Report on dependable software stated:

  > "Society is increasingly dependent on software. Software failures can cause or contribute to serious accidents that result in death, injury, significant environmental damage, or major financial loss. Such accidents have already occurred and without intervention, the increasingly pervasive use of software - especially in arenas such as

transportation, heath care, and the broader infrastructure - may make them more frequent and more serious.” [1]

The problem of software dependability is exacerbated by a pervasive lack of evidence about both the incidence and the severity of software failures. Software failures are often not reported and the severity of those failures that are reported is often understated.

To address this problem, future SQA teams will need to learn new techniques that help address the issue of dependability and evidence. Safety cases are an example of one such technique that will become more common in the next decade.

A safety case [2] presents high-level arguments that a system will be acceptably safe in a given context. For example,

- **High Level Arguments**

  These arguments represent an explanation of how available evidence can be reasonably interpreted as indicating acceptable safety – usually by demonstrating compliance with requirements, sufficient mitigation and/or avoidance of hazards etc.

- **Supporting Evidence**

  This evidence often includes results of observing, analyzing, testing, simulating and estimating properties of a system that provides fundamental information from which safety and dependability can be inferred.

Safety cases have been effectively used in air traffic control, passenger rail transit, and radioactive waste

disposal and storage projects. In the next decade, the use of safety cases will expand to other safety-critical software projects.

Read more about developing safety cases and look at some actual examples of safety cases

Pervasive Issues Facing the Profession

There are many issues facing the SQA profession that are pervasive. The two at the top of my list are:

- **SQA needs to be recognized as a discipline unto itself, just as software engineering was recognized back in the 1980s**

  It's time that universities and colleges offer degrees or specializations in **software quality assurance.** Currently, certificates offered by organizations such as the American Society for Quality (ASQ) and the Quality Assurance Institute (QAI) are woefully inadequate because they lack a foundation in software engineering.

  The software development community needs to pressure universities and colleges to create new degree programs and offer specializations to existing degree programs to address this pressing need.

- **SQA is a lot more than just testing**

  SQA professionals need to realize that the value of SQA goes way beyond testing. We need to understand the **SQA Umbrella** and take a more proactive role in all aspects of software development projects that affect quality.

The Bottom Line....

SQA is a vital discipline that must continue to be an essential part of software development projects. As software continues to expand into areas that are safety-critical (medical devices, transportation, nuclear power, etc.), the role of SQA in the next decade clearly needs to ratchet up to meet these challenges.

Prof. Edsger Dijkstra observed over 40 years ago:

> "The dissemination of knowledge is of obvious value — the massive dissemination of error-loaded software is frightening." [3]

'Til next time...

Monthly Morsels

Every month in this space, you'll find additional information related to this month's topic.

References

1. Jackson, D., et. al., *Software for Dependable Systems - Sufficient Evidence?* National Research Council, National Academies Press, 2007.

2. Kelly, T., "Assurance Cases, Argumentation and Patterns" High Integrity Systems Engineering Group, Dept. of CS, Univ. of York, UK.

3. "Software Engineering", Report on conference sponsored by NATO Science Committee Garmisch, Germany, Oct 7-11, 1968.

Calendar

Every month you'll find news here about local and national events that are of interest to the software community…

- **Software Quality Calendar**

  There are many organizations that sponsor monthly meetings, workshops, and conferences of interest to software professionals. **Find out what's happening...**

- **Workshops Offered by Software Quality Consulting**

  Software Quality Consulting offers workshops in many topics related to software process improvement. **Get more info...**

About SQC

Software Quality Consulting provides consulting, training, and

auditing services tailored to meet the specific needs of clients. We help clients fine-tune their software development processes and improve the quality of their software products. The overall goal is to help clients achieve Predictable Software Development™ – so that organizations can consistently deliver quality software with promised features in the promised timeframe.

To learn more about how we can help your organization, **visit our web site** or **send us an email**.

I hope this newsletter has been informative and helpful. Your comments and feedback are most welcome. **Send me your feedback...**

Thanks,

Steve Rakitin

info@swqual.com

Software Quality Consulting Inc.

Steven R. Rakitin
President

- Consulting
- Training
- Auditing

Phone: 508.529.4282
Fax: 508.529.7799

www.swqual.com
info@swqual.com