Workshop Summary of the 2015 Workshop on Smart Software Strategies:

15 Years after Y2K – Everything Old Is New Again (SMART 2015)

Dennis Mancl
Alcatel-Lucent
Murray Hill, NJ, USA
dmancl@acm.org

Steven D. Fraser
Innoxec
Santa Clara, CA, USA
sdfraser@acm.org

Bill Opdyke
JP Morgan Chase
Chicago, IL, USA
opdyke@acm.org

Abstract

Are there any good lessons that software people can learn 15 years after the Y2K crisis?

We live in a much more software-dependent world today, and the next generation of technical innovations may have some technical risks that have worldwide consequences. The 1990s was the most recent massive effort to improve and modernize software – and we might look to the past to explore some technical and management approaches that will prepare us for the "smart technology" wave.

As was the case in the 1990s, every company, every industry, and every country will need to be concerned with the potential risks in our software-driven world of the future: to better address software requirements, design, coding, and testing of our smart applications and smart support software. Are there some valuable technical and management ideas we can use again?

Categories and Subject Descriptors K.6.3 [**Software** Management]: Software Engineering; K.4.4 [**Electronic Commerce**]: Security.

General Terms Management, Reliability, Security.

Keywords Smart technologies; software risks.

SPLASH Companion'15, October 25–30, 2015, Pittsburgh, PA, USA
ACM. 978-1-4503-3722-9/15/10...
http://dx.doi.org/10.1145/2814189.2885248

1. Software Strategies: Discussion

How will Smart Software affect the future of software development? This is the primary question to be explored in this workshop. The workshop participants focused the discussion and conclusions on four fundamental questions:

- What is the "core knowledge" that software professionals need today?
- How should we change the teaching process to train good developers?
- What is the potential impact of Internet of Things (IoT)?
- How can we promote information sharing and effective root cause analysis?

For each of these four questions, there are no simple answers. On the other hand, there are some technology and training issues that will certainly become more important in the coming years.

"Core knowledge" for professionals must include "soft skills" and "critical thinking skills," not just the standard set of software design and coding practices.

The teaching process must go beyond the standard books and programming exercises: we need to have constant cross-pollination between academia and industry.

Internet of Things is an application ecosystem filled with risks. It is unforgiving to inexperienced or careless developers. Basic design mistakes might compromise user safety, security of subscriber data, system reliability, and the trust of the customer. The average proficiency level in the areas of software design and software testing need to be better than today.

Industry-wide feedback is an important practice to enable the quality level of Smart Software to improve over time. We might reuse some ideas from the aviation industry to collect more information about issues and problems.

2. Core Knowledge for Professionals

The top knowledge areas for doing effective development of complex distributed systems are not "coding skills." The designers of the next generation of Smart Software must have the skills to work with complexity. What is needed? Training in problem solving skills, knowledge of probability and statistics for analyzing performance and system dynamics, and experience with techniques for design abstraction.

Smart Software will be part of complex multi-vendor systems, so training in techniques for performing "needs analysis" to document user requirements, techniques for dealing with customers, and team management practices -- these are the key soft skills for product development. Most smart components will need to interact in a standard way, so future software designers must be familiar with industry standards in some key problem domains to help promote interoperability.

Also, because the role of Smart Software in society is important, training and knowledge of ethics is central, as well as economic tradeoffs in design and development.

Finally, influence skills will become more important. There will be more cross-industry teams working together to build end-to-end solutions. Some of these skills are documented in books such as *Influence: Science and Practice* by Robert Cialdini and *Fearless Change* by Linda Rising and Mary Lynn Manns.

3. Teaching Software Design Skills in the Internet Search Era

An increasing fraction of the "knowledge" of a software professional is detailed facts that are found through an Internet search… and this is not always an effective way to build a community of good designers. The biggest risk of "design by copying code from stackoverflow.com" is that an inexperienced developer can't tell if he/she is copying a good example or a bad example. Also, blindly copying code is a poor way to build experience: the learning process for any new technology involves both hard work and experimentation. The learning process must include some real "thinking" which will help the learner integrate the "hard-won lessons" into his/her working set of concepts and techniques.

Training approaches need to incorporate "building" as a central activity… and this will be most effective if there is a continuing partnership between academics and experts in industry. Academics need to do some industry work if they want to be effective educators.

4. Issues in the Internet of Things

The two biggest issues discussed in the workshop: data ownership and the transparency (or lack of transparency) of current and future IoT systems. If a set of IoT products and services have data ownership policies that are too vague or data privacy controls that are poorly implemented, these products will begin to lose the trust of customers. In order for network-enabled services to be successful, users must have confidence that their private information remains private. There will be trouble if the vendor and other third parties might assume that they own their customers' data and they can do anything they like… there will eventually be a backlash against these kinds of business practices. Smart vendors need to give customers a simple view of what data is being shared, and the view needs to be easier to use than today's home router log files.

5. Reporting Failures and Issues

In the United States, there is an Aviation Safety Reporting System (ASRS – http://asrs.arc.nasa.gov) that is managed by NASA. It is a public database that has been a key input for improving aviation safety and reliability. The ASRS program gives pilots an extra incentive to report: the person reporting incidents will be immune from FAA civil penalties for minor rule violations. The FAA considers reports of minor and inadvertent aviation safety incidents to be "indicative of a constructive attitude."

In the Smart Software world, it might be a good idea to create a similar public database. Sharing this data will improve smart software development practices industry-wide. It will enable developers to track down software interaction issues in a complex software ecosystem – to improve design, testing, and training.

There must be some economic and legal incentives for companies to contribute data, because most companies are reluctant to share this information. Data about software defects could have a short-term negative impact on sales and might be used in consumer legal action, but cross-industry analysis of design defects and blunders will have a positive impact on end-to-end quality.

6. Post-workshop Poster

A post-workshop poster summarizing the most significant ideas shared and questions generated during the session is posted on the workshop website:
http://manclswx.com/workshops/splash15/index.html