Panel

Trade-offs in Software Design and Delivery

Steven Fraser
Cisco Research Center
Cisco Systems, San Jose
sdfraser@acm.org

Richard Gabriel
IBM Research
Redwood City
rpg@dreamsongs.com

Gail E. Harris
Web Development Manager
and Architect
TV Ontario, Toronto
gail.e.harris@gmail.com

Ricardo Lopez
Software Architect and Consultant
San Jose
rjlopez@acm.org

Dennis Mancl
Distinguished Member of Technical Staff
Alcatel-Lucent, New Jersey
dennis.mancl@alcatel-lucent.com

William Opdyke
Architecture Lead
Corporate Internet Group
JP Morgan Chase, Chicago
opdyke@acm.org

Abstract

There are many design and delivery trade-offs that engineers face in creating or evolving software systems. Challenges in accelerating delivery, offering more features, providing better more reliable systems, or managing costs – whose optimization are just some of the hurdles that contribute to system success (or failure). This panel will discuss the heuristics of trade-offs, the inherent risks – and plans to build on the success of the 2012 SPLASH workshop "What Drives Design".

Categories and Subject Descriptors
K.0 Computing Milieux

General Terms Design, Experimentation, Standardization

Keywords Innovation, Creativity, Design Trade-Offs

1. Steven Fraser

STEVEN FRASER joined the Cisco Research Center as Director in July 2007 with responsibilities for fostering university research collaborations, managing PhD recruiting, and nurturing technology transfer. Prior to joining Cisco Research, Steven was a Senior Staff member of Qualcomm's Learning Center in San Diego, leading software learning programs and creating the corporation's internal technical conference (the QTech Forum). Steven held a variety of technology strategy roles at BNR and Nortel including: Process Architect, Senior Manager (Disruptive Technology and Global External Research), and Advisor (Design Process Engineering). In 1994 he spent a year as a Visiting Scientist at the Software Engineering Institute (SEI) collaborating with the "Application of Software Models" project on the development of team-based domain analysis (software reuse) techniques. Fraser is the Panels Chair for XP2013 and the Publicity Chair for ESEC 2013. He was the Corporate Support Chair for OOPSLA'08 and OOPSLA'09. He was the Tutorial Chair for XP2008 and the Tutorial Co-Chair for ICSE'09. Fraser holds a doctorate in EE from McGill University in Montréal – and is a senior member of the ACM and the IEEE.

SPLASH'12, October 19–26, 2012, Tucson, Arizona, USA.
ACM 978-1-4503-1563-0/12/10.

2. Richard Gabriel

RICHARD P. GABRIEL received a PhD in Computer Science from Stanford University in 1981, and an MFA in Poetry from Warren Wilson College in 1998. He has been a researcher at Stanford University, company president and Chief Technical Officer at Lucid, Inc., vice president of Development at ParcPlace-Digitalk, a management consultant for several start-ups, a Distinguished Engineer at Sun Microsystems, and Consulting Professor of Computer Science at Stanford University. He is a researcher at IBM Research, looking into the architecture, design, and implementation of extraordinarily large, self-sustaining systems as well as development techniques for building them. Until recently he was President of the Hillside Group, a non-profit that nurtures the software patterns community by holding conferences, publishing books, and awarding scholarships. He is on Hillside's Board of Directors. He helped design and implement a variety of dialects of Lisp. He is author of four books ("Performance and Evaluation of Lisp Systems," MIT Press; "Patterns of Software: Tales from the Software Community," Oxford University Press; "Writers' Workshops and the Work of Making Things," Addison-Wesley Press; and "Innovation Happens Elsewhere: Open Source as Business Strategy," Morgan Kaufmann), and a poetry chapbook ("Drive On," Hollyridge Press), with two books of poetry in preparation: "Leaf of my Puzzled Desire" and "Drive On." He has published more than 100 scientific, technical, and semi-popular papers, articles, and essays on computing. He has won several awards, including the AAAI/ACM Allen Newell Award. He is the lead guitarist in a rock 'n' roll band and a poet.

Design in the future will have two distinct and mutually contradictory challenges. Remember: the future.

First, all the programs that can be written by a single person or a team working together have already been written, and every interesting new program cannot be subject to whole-system design. Neither requirements nor design will be consistent. Every designer will be limited to a narrow part of the program's interface or to its interstitial glue. In the past, design was like creating Esperanto - control of every aspect - while now design is like adding a new slang phrase to English - something akin to "Shatner texting." One way to do this is like Siri: a small interface on the iPhone designed by designers (using guidelines from Apple and subject to their approval), plus a raft of code in the cloud (put together over a decade or more by a team originally scattered and now long gone). This could be called iceberg architecture.

Second, all the crap that goes with finding, acquiring, installing, maintaining, upgrading, and using software and that is not about the actual task (let's call it) the software's user wants to accomplish has to be scraped away from view, must be invisible to the buyer, user, and everyone on that end of the whole transaction - this is a designer's task. Some have called this "ready-to-hand." It's a kind of whole-system design. More recently some corporations have taken to calling this "consummability." Siri is a way to accomplish consummability by hiding all the crap in the cloud, but can all software be cloudy?

3. Gail E. Harris

GAIL E. HARRIS was recently appointed Web Development Manager and Architect at TVOntario (TVO), the Province of Ontario's public educational media organization. Gail is responsible for all technical aspects of TVO's web and mobile presence, including long term strategy and development methodologies. Prior to joining TVO, Gail was a Principal and co-owner of Instantiated Software, a company that applied agile methodologies and open source technologies to successfully deliver custom applications to start-up companies. Previous to Instantiated, Gail worked for several larger organizations including the Department of National Defence, and Deloitte Consulting. For the past fifteen years Gail has been a regular contributor to SPLASH/OOPSLA. Gail was the OOPSLA Conference Chair in 2008.

Not too long ago, on a modest sized system that had been running for a few years, a customer requested a seemingly simple change to the text on a certain web page. The complexity, and hence the design challenge and trade off, showed up while doing the analysis. While the text needed to vary according to the data being displayed, more importantly, the web page in question was displaying an invoice. In addition to the required text change, there was also an underlying constraint that an old invoice needed to be presented exactly as it would have appeared at the time it was issued. A backward compatibility requirement. Backward compatibility may not be a new topic, nor a cool topic. It does however force designers to think strategically about the compromises they make. Should I put time and effort (money) into programming the strategy pattern or the facade pattern? Can I limit compatibility to no more than X major historical releases? How will I maintain code readability and repair-ability?

In this particular example something interesting occurred. Near the very beginning of the project the designers had decided that all invoice data would be retrieved from the business model objects and copied into a completely separate set of read only database tables, allowing for historical trend analysis. These tables would be queried to display invoices, not the core model tables. Furthermore, the text in question existed in the invoice template in the view layer. Two main options were considered:

- detect the version in the view layer and generate the appropriate text for invoices of different ages
- add a database field for the text, modify the view to display it, modify the core model to generate the text based on version, and populate all the old invoices.

The trade-off here is between effort (cost) and separation of concerns that keeps business logic separate from view logic. After consulting with the customer the designers chose the former option, because the nature of the business suggested that it would be highly unlikely to have another change. This meant that the changes would not require backward compatibility of the core classes; the programming changes were isolated in the view layer. The less nice observation is that the view layer now has a smell: date pollution. In a few places the code includes some conditionals about the version needed for the invoice being presented. The residual design dilemma is how much effort to put into removing that smell, if it's even possible.

The views expressed in this position statement are those of Gail E. Harris and do not represent those of her employer.

4. Ricardo Lopez

RICARDO LOPEZ is a software architect and consultant. Formerly he was a Principal Engineer at Qualcomm CDMA Technologies and adjunct to the Office of the Chief Scientist at Qualcomm. He was responsible for software architecture, software process, and sometimes Just Good Old Fashioned Software – AKA Code. Architecting and designing Software for over thirty-five years (too old for Google), he has been an evangelist for OO technology for the last twenty-five years and he has the arrow heads to prove it (time to become an early adopter of the next great orientation)...

5. Dennis Mancl

DENNIS MANCL works for Alcatel-Lucent, where he is involved in applying software modelling approaches, agile development practices, and legacy software development techniques to the development of large telecom systems. He has worked with technologies from C++ to UML to Scrum, with a preference for simple designs, simple tools, and simple metrics.

Software design and software delivery are difficult. The design process, which builds up the structure of a proposed solution to a real world problem, requires a combination of experience and creativity from the designer. The delivery process, which reshapes the design and its implementation to closely fit the current customer expectations, requires patience and attention to detail.

How can we coordinate the design process and the delivery process, especially in a world of rapidly evolving customer needs? A number of design approaches and process models have been proposed over the years, from Object Oriented Design to Extreme Programming, with some success. Maybe any organized design approach will work, as long as the developers believe in it.

6. William Opdyke

BILL OPDYKE has spent much of his career focusing on the technical and organizational issues related to transitioning advanced software technologies and software engineering techniques into product development. He is currently on staff at JP Morgan Chase. Previously, at Motorola, he was part of an advanced technology team focusing on home networking related middleware and on techniques for improving productivity and reducing costs of software developments. While at Bell Labs, he was technical lead on several advanced development projects where he gained a keen appreciation for the challenges in leveraging emerging technologies and in extending existing products to meet emerging market needs. He also spent several years as a faculty member at North Central College. His doctoral research at the University of Illinois focused on object-oriented refactoring (supporting the process of change to object-oriented software).