Panel

Architecture in an Agile World

Steven Fraser
Director
Cisco Research Center
Cisco Systems, San Jose

Ethan Hadar
VP Research
CA Labs, CA Inc
Haifa, Israel

Irit Hadar
Head, Software Architecture Laboratory
Management Information Systems
University of Haifa, Israel

Dennis Mancl
Distinguished Member of Technical Staff
Alcatel-Lucent
New Jersey

Grenville (Randy) Miller
Architect and Author
Microsoft
RTP, NC

Bill Opdyke
Distinguished Member of Technical Staff
Motorola
Schaumburg

Abstract

This panel will explore the apparent dichotomy of agile-centric and architecture-centric approaches to software development. The panel will address questions of interest that include: How are architecture practices applied in a world where 'agile' must co-exist with more traditional sequential processes and the scale and scope of legacy systems? and – What are the organization structures, tools, methods, and education/training strategies that can be used to maximize the delivered value and reduce the cost of system ownership? This panel is intended to build on the outcomes of OOPSLA 2009 workshop with the same title.

Categories & Subject Descriptors:
D.2.11 Software Architectures
K.0 Computing Milieux
K.4.3 Organizational Impacts

General Terms: Design

Keywords: Software architecture, agile development, design

1. Steven Fraser *(panel impresario)*, *sdfraser@acm.org*

My premise is that the larger the system, the more necessary the architecture. To enable economies of scale and scope – an appropriate architecture is absolutely essential. This is not only true in the software development phase, but also in the application space. Today's "now" world demands inter-operability and rapid ease-of-use. In most ways an "agile world" is no different from a "sequential world." The exception is that of the requisite feedback cycle of agility (a sequence has been iterated at least once) where the benefit of single (or double) loop learning may be leveraged and learning is captured by "architecture."

STEVEN FRASER is the Director of the Cisco Research Center in San Jose California with responsibilities for developing and managing university research collaborations. 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 also held a variety of technology and management roles at BNR/Nortel. In 1994 he spent a year as a Visiting Scientist at the SEI collaborating on the development of team-based domain analysis (software reuse) techniques. Fraser was the Corporate Support Chair for OOPSLA'08, 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 member of the ACM and a senior member of the IEEE.

2. Ethan Hadar, *ethan.hadar@ca.com*

Dedicated architecture and design activities usually involve some discussions about "how low should we get?" Whether the design includes components, integrations, platform selections, or non-functional requirements, the participants are always eager to test, try, build and see that it works. Architectural debates are over-detailed and abstraction is not maintained.

While this may work with developing code in an agile approach, the effect on large-scale systems development might have a different result. In large teams, or ones that have service level agreements' needs, the team is required to first "slice and dice" the system into manageable parallel assignments, and bound them by contracted integration points. Lack of an architecture roadmap and clear understanding of where we need to go, will constitute a spontaneous and opportunistic architecture, instead of being driven by an architecture-centric approach. Thus, we might finish the project, but not as intended. Can large components be assembled in a structured agile manner, rather than opportunistically? If the APIs will change as well as the components' responsibility due to lack of clear up-front understanding – can we still provide a solid progression, on a firm committed timeline bounded schedule, and with minimal correction iterations?

Architecting a set of deployable and testable components with well-defined interfaces in an ambiguous environment requires architecture planning and related activities to be agile. Some architectural decisions (frameworks, libraries, platforms, etc.) are resistant to change while others may require frequent adjustment (such as components structure and responsibility).

Our position is that it is important to maintain separation between architecture layers (external integration, functional, system and common components), as well as defining a reference (future) and implementation (next release) architecture. The essence of an agile architectural approach is to provide a connecting roadmap between architectures in combination with a non-functional assessment of evolving needs.

ETHAN HADAR is a SVP for Research at CA Labs, and Distinguished Engineer at CA Inc. His responsibilities include leading strategic research and corporate guidance on Cloud Computing, Architecture, domain specific modeling, and ITIL, in collaboration with CA R&D groups, Customers, CA's Executives and academia. Ethan's recent activities include architecture implications in agile environments, ITIL automations in cloud arena, and integration-oriented domain specific languages and modeling (DSL/M). Prior to joining CA, Ethan was the principal architect at Mercury Interactive, now HP Software, where he developed new methodologies in software engineering, service oriented architectures, and object oriented technology. Ethan has numerous patents and publications, and he served as a member of the faculty at the Netanya Academic College and as adjunct faculty at the Technion, Israel Institute of Technology. He holds a Ph.D. from the Department of System Analysis and Operations Research at the Technion.

3. Irit Hadar, *hadari@mis.haifa.ac.il*

Human aspects influence, and are influenced by, the construction of architecture in different approaches. From the human perspective, there is a spectrum of software development activities, some of which do not require an architecture control, and some do. There is a major difference between the architecture needs and impact in small teams, developing small programs *versus* large-scale software systems, e.g. enterprise solutions. Realizing this difference, we need to be able to identify the different cases, how exactly they differ, and whether some similarities between them exist. Thus, we need to ask: Is there a continuous spectrum of solutions' scale, where we can detect the area (or boundary) in which the approach needs to be changed? Are there architecture artifacts and activities that all methodologies must maintain?

It is believed, justly I think, that agile methods have an important role in dealing with the difficulties and challenges of software development that are related to our human nature. These include, for example, psychological issues such as cognitive overload when handling highly complex systems' development and maintenance, or the desire to see the code running as early as possible in the development process; and social issues that stem from the necessity for collaboration between developers, which are magnified in complex and large systems' development.

Most of the principles of Extreme Programming are oriented towards making things simple for the human mind as well as supporting collaboration and human communication. Let's take the metaphor principle as an example. The benefit of using metaphors in any science is making abstract concepts more concrete and familiar, thus ease the cognitive effort and support communication when discussing them. However, using metaphors can only get you so far. At some point the metaphor no longer spans the entire essence of the concept it replaces. At this point we need, in our case, an expert architect that is able to elevate the discussion to the abstraction level required, in order to achieve a true quality solution. The challenges we face in this example are to identify this point, and find a way to handle the architecture differently and appropriately from this point onwards.

IRIT HADAR is a faculty member at the Department of Management Information Systems at the University of Haifa. She is the head of the Software Architecture Laboratory at CRI (Caesarea Rothschild Institute for Interdisciplinary Applications of Computer Science). Her main research area is cognitive aspects of software architecture, design and analysis. In particular, she focuses on cognitive processes, difficulties and conflicts in software development, applying cognitive psychology theories as analysis frameworks; visual models in software development and their influence on developers' perceptions; and the influence of different aspects of software engineering – economic, social and cognitive aspects – on the final quality of the developed software.

4. Dennis Mancl, *mancl@alcatel-lucent.com*

Every system has an architecture. It may be a much disorganized architecture, but it is an architecture nonetheless. In an agile world, it is just as important to invest in architectural planning as before.

There are four major reasons why you need a documented architecture: An early architectural description is useful for the search in the search for external that you might reuse – a good way to reduce development effort and the cost of your final product. A documented architecture makes more complete product testing possible. For a long-lived software product, an architecture description helps new development team members to learn about the structure of the software, reducing errors and development churn. There may be modules in your software product can be reused in another context, and an architecture description is necessary to describe where the modules will work reliably.

Agile development processes must include tasks and artifacts to support some amount of architecture planning and evolution. In an agile world, there may be many small and short-lived systems where it is reasonable to just "let the architecture emerge" during the development process. Exploratory software and throw-away prototypes can get by with little architectural planning. But for most real product software, some form of lightweight architecture documentation and ongoing architecture evolution activities need to be parts of the development lifecycle.

DENNIS MANCL is a Distinguished Member of Technical Staff at Alcatel-Lucent in Murray Hill, NJ. His interests range from software requirements practices to legacy code transformation techniques. Dennis has been an internal consultant on OO design within Alcatel-Lucent and AT&T, with considerable experience in assisting software project teams with design patterns, requirements modeling, and reengineering existing software.

5. Granville Miller, *Randy.Miller@microsoft.com*

Architecture is a necessary component of software system development. This is, of course, evident for mid-sized to large systems but also necessary for smaller systems. Ward Cunningham defined architecture as "the moment in the development of a system when you have to step back, look at, and structure the system for further development". The story that Ward uses to support his definition describes a system created by a single developer. In circumstances where there are hundreds of developers, there are many of these moments. If nothing is done when these moments call for action, the resulting systems will be poorly architected. Systems that are poorly architected or without architecture fall apart rather rapidly as they collapse under the weight of their own functionality.

If architecture is so necessary, why are we debating its merit in the agile community? Many software professionals confuse architecture with "Big Design Up Front" (BDUF). Many a software development project failed to produce anything as "an architecture" consumed all of its development time. Moreover, many of these architectures were found to be invalid as the development team learned new things in the course of development. The fact is, architectures do not need to be developed completely before the project begins. Instead, they are living elements that change with the project.

A good architect keeps their developers from "painting themselves into a corner" as they build the system. Architects do this through experience and earned respect. The modern software (solutions) architect is a vital part of the project, delivering functionality. Agile processes are aids to the architect as they force the team to make hard decisions and surface risks. But the agile architect knows when to follow a process and when to adapt the process to fit the needs of a project. Ultimately, they know that the delivery of a maintainable system is the mark of a successful project, not strict adherence to any process.

GRANVILLE MILLER is an architect working on high-risk projects for Microsoft. He is the co-author of Advanced Use Case Modeling and A Practical Guide to Extreme Programming. He brings 22 years of software development experience in the delivery of software applications. He has been actively promoting modeling, software development technology, and software process for over a decade in various public forums. He has an Advanced Certification in UML 2.0.

6. Bill Opdyke, *opdyke@acm.org*

Is there really a dichotomy between agile-centric and architecture-centric approaches to software development? My experience has been that the dichotomy is more fiction than fact. Sure, I've sometimes seen software engineers practice a form of anarchy and seek to justify it by labeling it agility, while others live in a world of arcane abstractions, produce few usable results and justify their work by labeling it architecture. However, more often I've had the pleasure to work with outstanding agilistas and outstanding architects, and I've seen a common passion and focus (among these agilistas and architects) on providing value to end users, accommodating change, collaborating with other team members and helping make innovation happen.

When it comes to interworking with legacy products (and legacy processes), outstanding architects have much to offer to agile projects. I've seen agile projects struggle to communicate their progress and effectively collaborate with companion projects that are based on more traditional development processes. The outstanding architects I've worked with have been skilled in abstracting, adapting and communicating – important skills required to translate between different development cultures and to achieve collaboration.

Correspondingly, I've seen how outstanding agilistas can bring the architecture and development roles closer together in ways that build shared accountability and shared purpose, and enable the overall project to accommodate change in ways that provide both near and longer-term value. I've spent a couple decades applying and training people in structures, tools and strategies related to architecture and to agile methods – and will share some of my insights in these areas at the workshop and panel discussion. In summary: outstanding architects and outstanding architects have much in common, and have much to gain from collaborating with each other.

BILL OPDYKE has spent much of his career in system and software architecture roles at Bell Labs/AT&T/Lucent and more recently at Motorola – on projects that successfully applied agile techniques – before they were called "agile". His doctoral research led to the foundational thesis on refactoring, an important element of agile software development. Bill has taught agile and architecture related concepts in both academia and industry.