



Lightweight and Flexible Emerging Trends in Software Architecture from the SATURN Conferences

Michael Keeling

Software architecture is a rather young, fast-moving discipline. It's beneficial to reflect once a year about what has been achieved so far, and the SATURN conference has become one of the premier places to do so. Michael Keeling did an excellent job as the SATURN 2014 program chair; in this article he distills four of the most important software architecture trends that were highlighted during the conference: architecting for DevOps, flexible designs, lightweight architecture design methods, and renewed interest in software architecture fundamentals.

—Cesare Pautasso and Olaf Zimmermann, department editors.



OVER ITS 10-YEAR HISTORY, the annual Software Engineering Institute (SEI) Architecture Technology User Network (SATURN) conference has become a barometer for the ever-evolving software architecture climate. SATURN is a great conference, and it was my privilege to be the SATURN 2014 technical chair. Here, I summarize software architecture trends I saw emerge during the conference and give a glimpse of the future based on the current SATURN 2015 technical program. (For a brief rundown of SATURN 2014, see the sidebar.)

Four Trends

If one idea stood out at SATURN 2014, it was that promoting business agility requires sound architecture design. Discussion in the hallways between sessions

wasn't about whether architecture and agile worked together but more about how best to achieve agility through architecture. With this as the main theme, I noticed the following four supporting ideas throughout the conference.

Architecting for DevOps

With nine sessions covering DevOps-related topics, DevOps seceded everywhere. DevOps is an emergent software development approach that blends the traditional roles of operations and software development to increase an organization's ability to deliver business value. Groups practicing DevOps take direct responsibility for the whole user experience. This includes everything from infrastructure and tools to development, testing, and deployment, and much else besides.

SATURN 2014 BY THE NUMBERS

Hosted by the Software Engineering Institute (SEI) in collaboration with *IEEE Software*, the 2014 SATURN (SEI Architecture Technology User Network) conference ran from 5 to 9 May in Portland, Oregon. Here's a brief summary of what took place:

- 198 software architects, software developers, and thought leaders attended.
- 27 countries were represented.
- 33 experience report presentations shared stories and lessons across three broad themes: Technical, Methods and Tools, and Leadership and Business.
- 12 tutorials and courses covered topics such as big data, software architecture design analysis, architecture hoisting, DevOps, architecture katas, and architecture-centric agile practices.
- 4 participatory sessions gave attendees the chance to immediately practice new methods.
- 4 keynotes were delivered by Bill Opdyke (JPMorgan Chase), Joe Justice (Scrum Inc. and Team WIKISPEED), Diane Marsh (Netflix), and Jerome Pesenti (IBM).
- 1 panel explored the role of technical debt from the perspective of architecture. The moderator was George Fairbanks (Google); the panel comprised Jeremy Carrière (Google), Philippe Kruchten (University of British Columbia), Robert L. Nord (SEI), Michael Keeling (IBM), and Rebecca Wirfs-Brock (Wirfs-Brock Associates).
- 1 Open Space event was facilitated by Diana Larsen (FutureWorks Consulting), with numerous attendee-led sessions and serendipitous encounters among “butterflies.”

Nearly all SATURN 2014 keynotes and experience reports are available online as slides, video, or both.^{1,2}

References

1. *SATURN 2014 Presentations*, Software Eng. Inst., 2014; <http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=89532>.
2. *SATURN: A Software Architecture Community*, Software Eng. Inst., 2014; <http://bit.ly/saturn2014-video>.

Many of the benefits promised by DevOps can be achieved only when the system's architecture promotes DevOps values and enables specific practices. So, DevOps values must be promoted in the architecture similarly to any desirable quality attribute. Much like security, avail-

ability, scalability, and other quality attributes, DevOps can't simply be bolted on as an afterthought.

The importance of architecting for DevOps was evident throughout the conference, the most obvious being during Diane Marsh's keynote describing how Netflix supports

continuous delivery. One way Netflix achieves this is through an automated test suite affectionately called the Simian Army. Tools such as Chaos Monkey, a program that randomly kills running application instances in production, codify desired quality attributes in a way that diagrams and prose can't. For example, Netflix's arsenal of automated tools provides practical tests for evaluating how well the system achieves certain runtime quality attributes such as availability and scalability.

In many ways, Chaos Monkey and its companions provide a test-driven-like approach for some quality attributes in the architecture. Chaos Monkey clearly defines reliability and availability in the form of an executable test that continuously runs during normal business hours. A developer's ability to progress through the red-green-refactor cycle at the architectural level is powerful indeed.

Naturally, Netflix didn't arrive at this design overnight. The highly malleable and resilient Netflix architecture is the result of years of evolution and hard-earned lessons. If nothing else, the Netflix design serves as a shining example of what might be achieved when the organization, culture, and technology are aligned.

Flexible Designs

Netflix's experience is also a good example of the importance of flexible designs, and there were many other examples at SATURN. Flexible designs enable organizational agility. Here I discuss two of the more exciting examples from the conference.

Flexibility was essential to IBM's success in the 2011 *Jeopardy!* IBM Challenge. During his keynote, Je-

rome Pesenti shared the story of IBM's journey to create Watson, a cognitive computing system that defeated the best human *Jeopardy!* contestants. *Jeopardy!* is an American television game show that provides trivia clues as statements, to which contestants provide answers in the form of a question. Watson's victory was an important milestone in AI. However, Watson might not have been possible without the highly flexible, modular, and parallelizable architecture the research team adopted.

Early on, the Watson team realized that no single algorithm would provide question-answering accuracy sufficient to beat the best *Jeopardy!* contestants. The solution lay in integrating multiple algorithms to create compounding improvements in answer accuracy. A highly modular architecture that promoted rapid experimentation was critical to the team's success. By the time Watson was ready to compete, the relatively small 20-person team had conducted more than 5,500 independent experiments over three years, totaling more than 11 million CPU hours.¹ On the basis of these experiments, the team picked the best combination of algorithms for the competition.

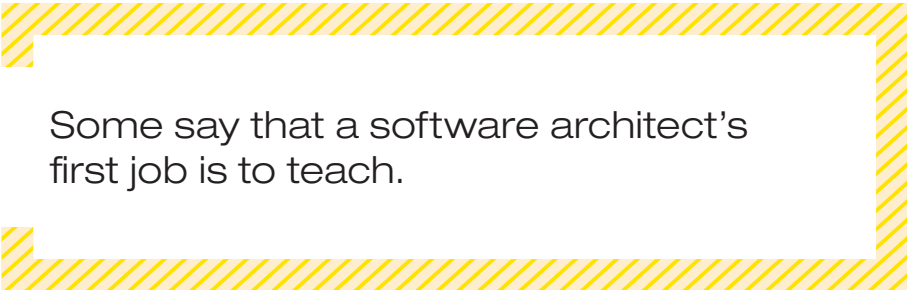
As Joe Justice's keynote discussed, flexibility was also essential to Team WIKISPEED's success in the Progressive Insurance X Prize competition. In less than three months, a team of volunteers designed and constructed a car that could achieve over 100 miles per gallon. Justice attributes his team's success to its use of agile methods and object-oriented design. Neither practice is characteristic of automotive manufacturing.

The WIKISPEED garage is built around common agile practices such as pairing, test-driven develop-

ment, the definition of "done," and Scrum. This lets new teammates quickly pick up new knowledge, apply it, and contribute to vehicle construction. Justice calls the method Extreme Manufacturing, in homage to the method's inspiration, Extreme Programming.

Likewise, WIKISPEED cars are designed to be modular from the beginning. As with the Watson team, modular design lets Team WIKISPEED quickly iterate designs. In fact, WIKISPEED has developed a

tecturally evident coding styles, lightweight representations, essential modeling skills, and lean design methods have appeared regularly in the SATURN technical program. Many presentations covering these topics have won the conference's *IEEE Software* SATURN Architecture in Practice Presentation Award, including at SATURN 2014. This peer-voted award goes to the presentation that best describes lessons learned while applying architecture-centric practices.



Some say that a software architect's first job is to teach.

product line of high-efficiency vehicles, from trucks to sports cars, based on the same modular framework. It's amazing how the fit and function of agile and architecture become plainly evident when the same practices are applied in a setting other than software.

Although Watson and WIKISPEED might be extreme examples of how agile design can achieve extraordinary outcomes, the increasing business demands and complexity of many systems requires that architects think about agility first, regardless of the development methodology.

Lightweight Architecture Design Methods

Trimming the fat from software architecture methods has been a trending theme at the past few SATURN conferences. Topics such as archi-

The 2014 award went to Will Chaparro's and my talk, "Facilitating the Mini-Quality Attributes Workshop."² I think our presentation won because it provided practical, actionable advice and because the mini-QAW updates the traditional Quality Attribute Workshop³ to address modern agile teams' concerns.

At the heart of the mini-QAW is a quality attribute taxonomy, a set of common quality attributes relevant to typical stakeholder concerns, classified for easier consumption. Our experience was based on using a taxonomy for enterprise search applications; some generic taxonomies are available in the technical report *Quality Attributes*.⁴ By using a taxonomy to enrich facilitation options, we could skip or reduce several steps in the traditional QAW so that the time

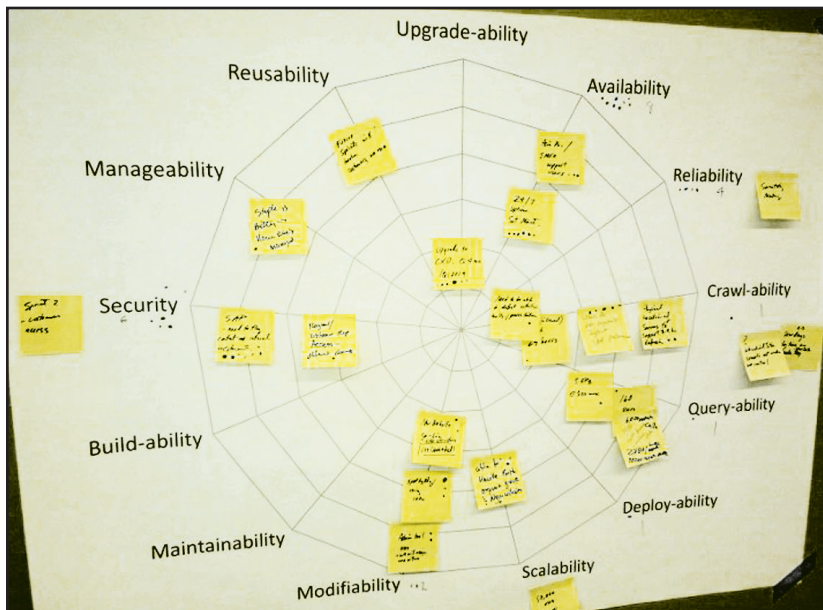


FIGURE 1. A quality attribute taxonomy used as a facilitation aid during a mini-QAW (Quality Attribute Workshop) with stakeholders. Sticky notes contain raw scenarios; dots indicate priority determined by a dot-voting exercise.

to complete a workshop decreases from a few days to a few hours, depending on the circumstances.

Mini-QAW facilitators can use a taxonomy directly in many ways. For working with experienced stakeholders, the taxonomy aids structured brainstorming—for example, placing sticky notes on a whiteboard clustered around taxonomic terms (see Figure 1). During a workshop, the facilitator can also “walk the web” with stakeholders by asking questions from a taxonomy-based questionnaire. In this way, the taxonomy provides a ready starting point and a concrete guide for facilitating the workshop. Such a questionnaire is an excellent accompaniment for inexperienced facilitators and stakeholders.

I think lightweight methods have proved popular at SATURN because they enable teams to learn fast, fail fast, change fast, and communi-

cate effectively. These factors are essential for self-organizing teams, which in turn enable better designs to emerge.

Renewed Interest in Software Architecture Fundamentals

None of the many forward-looking trends at SATURN 2014 would have been possible without a strong foundation in software architecture. Teaching the next generation of software architects seemed to be on many speakers’ minds. Some of my favorite talks involved either sharing deep knowledge in software architecture or novel approaches to teaching as a means of growing the next generation of architects.

One highlight was Ted Neward’s tutorial “Architectural Katas.” During an architectural kata, attendees work in small groups to architect a system from scratch, with the facilitator acting as the customer

stakeholder.⁵ Inspired by code katas (which were in turn inspired by martial-arts katas),⁶ an architectural kata creates a no-risk opportunity for software developers and architects, both experienced and novice, to practice software architecture design.

During our kata, teams explored business and architectural drivers and designed several views of a system that showed how the most important quality attributes might be achieved. As a conference organizer, I found it particularly rewarding to see participants applying ideas they had learned at the conference.

New research addressing longstanding ignorance was also great to see. “Approaching Security from an ‘Architecture First’ Perspective,” by Rick Kazman, Jungwoo Ryoo, and Humberto Cervantes,⁷ won the *IEEE Software* SATURN New Directions Presentation Award. This peer-selected award goes to the presentation best describing innovative ideas that advance the state of architecture-centric practice. I think this presentation won not because its revelations were necessarily groundbreaking but because Kazman and his colleagues directly addressed longstanding software architecture myths about promoting security. Their research was solid and presented in a way that was relevant to practitioners.

Their most important finding showed that the most effective way to promote security in a system is early adoption of a security framework such as Apache Shiro or Spring Security. In short, “delegating security issues to frameworks allows developers to devote their energy to application logic, increasing overall productivity.”⁷

Effectively sharing what we know about software architecture was the

focus of George Fairbanks' talk, "Teaching Architecture Metamodel-First."⁸ Fairbanks outlined five common obstacles to teaching software architecture, and strategies for overcoming them. For example, to remove abstraction as an obstacle, lessons with new architects should focus first on diagramming rather than analysis and employ plenty of hands-on exercises. Also, to overcome burgeoning architects' tendency to focus on the wrong details, show them how to start with a metamodel by creating the legend for their diagrams first.

Some say that a software architect's first job is to teach. Effectively communicating decisions about a software system's architecture requires that all stakeholders have a firm grasp on software architecture fundamentals. Knowing the fundamentals is important, and so is knowing how to teach them to others. It was great to see talks focused on teaching the teacher, a role every software architect inevitably fills at least sometimes.

SATURN 2015 Highlights

The 11th SATURN conference will be in Baltimore from 27 to 30 April 2015, with the return of both George Fairbanks (the SATURN 2012 technical chair) and me as the technical cochairs.

What trends will emerge from SATURN 2015? Although I don't want to speculate too wildly on what trends might emerge, some talks have already been accepted and offer a glimpse of what might come. Scheduled topics include reducing technical debt, cloud architectures, and microservices. Design thinking and similar reflective, adaptive design frameworks also appear to be trending in the submitted proposals.

DevOps is still present but doesn't seem as prevalent as in 2014, but we'll know more as the conference unfurls in April.

The SATURN 2015 keynotes will be by Mary Shaw, a professor of computer science at Carnegie Mellon University and winner of the US National Medal of Technology and Innovation; Mark Schwartz, Chief Information Officer for US Citizenship and Immigration Services; and Gregor Hohpe, Chief IT Architect at Allianz.

Finally, a special track consisting of sessions curated by invited thought leaders will cover a variety of emerging and classic topics. You'll likely recognize many of the speakers' names, including Simon Brown, Ariadna Font, Len Bass, Rebecca Wirfs-Brock, Jeff Patton, Joseph Yoder, and Jeromy Carrière.

For more information and archived presentations, visit www.sei.cmu.edu/saturn/2015. On behalf of the program committee, I look forward to seeing you at SATURN 2015! 🎧

References

1. D. Ferrucci et al., "Building Watson: An Overview of the DeepQA Project," *AI Magazine*, vol. 31, no. 3, 2010, pp. 59–79.
2. W. Chaparro and M. Keeling, "Facilitating the Mini-Quality Attributes Workshop,"

- presentation at the 10th SEI Architecture Technology User Network (SATURN) Conf., Software Eng. Inst., 2014; <http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=89553>.
3. M.R. Barbacci et al., *Quality Attribute Workshops (QAWs), Third Edition*, tech. report CMU/SEI-2003-TR-016, Software Eng. Inst., 2003.
 4. M. Barbacci et al., *Quality Attributes*, tech. report CMU/SEI-95-TR-021, Software Eng. Inst., 1995.
 5. T. Neward, *Architectural Katas*, Neward & Associates, 2012; <https://archkatas.herokuapp.com>.
 6. D. Thomas, *Code Kata*, blog, 2013; <http://codekata.com>.
 7. R. Kazman, J. Ryoo, and H. Cervantes, "Approaching Security from an 'Architecture First' Perspective," presentation at the 10th SEI Architecture Technology User Network (SATURN) Conf., Software Eng. Inst., 2014; <http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=89604>.
 8. G. Fairbanks, "Teaching Architecture Metamodel-First," presentation at the 10th SEI Architecture Technology User Network (SATURN) Conf., Software Eng. Inst., 2014; <http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=89518>.

MICHAEL KEELING is a software engineer at IBM and a member of the IBM Watson Explorer team. His professional interests include software architecture design, agile methods, and making awesome software. Keeling received a master's in software engineering from Carnegie Mellon University. Contact him at mkeeling@neverletdown.net.



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.



The podcast for professional developers is looking for hosts to interview some of the top minds in software engineering.

Contact bbrannon@computer.org for more information.

Sponsored by **Software**