Continuous Delivery – From Concept to Product: 
Trade-offs in Effectiveness and Efficiency?

Steven Fraser¹, Ismo Aro², Henri Kivioja³, Erik Lundh⁴, 
Ken Power⁵, Linda Rising⁶, Werner Wild⁷, and Rebecca Wirfs-Brock⁸

¹Innoxec, CA, USA
dsdfraser@acm.org
²Omenia, Finland
ismo.aro@omenia.fi
³Ericsson, Helsinki, Finland
henri.kivioja@ericsson.com
⁴Compelcom, Sweden
erik.l.lundh@gmail.com
⁵Cisco Systems, Galway, Ireland
ken.power@gmail.com
⁶Independent Consultant, TN, USA
risingl@tds.net
⁷University of Innsbruck, Austria
werner.wild@uibk.ac.at
⁸Wirfs-Brock Associates, OR, USA
rebecca@wirfs-brock.com

Abstract. The implementation and release of software products has progressed from a lengthy delivery cycle – the methodical sequential path of “big bang” waterfall product delivery – to the rapid iterative release cycle supported by agile practices. Recently, “continuous delivery” has emerged as a strategy to accelerate product availability. However, only a systematic automation of the build, test and deployment processes in concert with superbly coordinated teams of software practitioners and business partners makes this possible. However, trade-offs in the optimization of process may act to limit the innovativeness of product output. This panel will discuss approaches, challenges, risks, and strategies for using continuous delivery to competitive advantage.

Keywords: Agile · Waterfall · Automation · Processes · Innovation · Delivery

1 Steven Fraser (Panel Impresario)

Steven Fraser is based in Silicon Valley California and has served as an innovation catalyst with global influence for three Fortune 500 Companies as: Director, Cisco Research Center; Senior Staff, Qualcomm Learning Center; and Senior Manager, Nortel Disruptive Technologies and Global External Research. In addition to a year spent as a Visiting Scientist at Carnegie Mellon University’s Software Engineering
Institute (SEI) consulting on Domain Engineering (software reuse) processes and practices he has organized and delivered over 75 software engineering conferences, panels, keynotes, workshops, and tutorials.

Continuous delivery of software features, systems and services is a dream that may becoming reality. This panel will discuss challenges and results – observing and reporting on current best practices and pitfalls.

2  Ismo Aro

Ismo has an extensive track record in the corporate world as a change agent. He is enthusiastic to work with companies to improve their capability to produce customer value faster and drive them to be amazing places to work. Nowadays he works as an entrepreneur at Omenia.

Inventories are one of the biggest source of waste in software development. Companies that develop software accumulate immaterial inventories as feature requests in backlogs, in code waiting, in source control systems, or software packages waiting for deployment. We should systematically locate and reduce these inventories, by finding constraints and exploiting them.

3  Henri Kivioja

Henri has experienced and lived transformations in many corporations: moving from a two year release cycle to a more frequent delivery and deployment cadence. He has learned from experience with large telecom operators on what it means to move towards rapid value creation. Henri has been fortunate enough to be at the center of change as a project and program manager and more recently as a head coach. He has learned the fundamentals of large-scale change, feedback mechanisms and people. Henri is a popular public speaker in the Lean-Agile community.

Continuous Deployment should be based on market and business demand. Another dimension is that how (and if) we understand how it works. Conventional management treats this aspect as efficiency. In knowledge, work efficiency is a broader concept and sometimes misunderstood. We need to agree on some general terms and concepts. What is continuous? What is the purpose of R&D? How does the contractual situation affect value creation? How is culture impacting teamwork and way the work is improved? What is the relation between Continuous Integration, Continuous Delivery and Continuous Deployment? All in all the term “competitive advantage” needs to be rephrased in modern software and knowledge work, as tools and availability of different technologies is possible outside the industries’ entry barriers (regulation, cost, availability). This implies that advantage is created through people and culture – not solely with tools and frameworks.
4  Erik Lundh

Erik Lundh has more than 30 years’ experience in product-technical-software development beginning as an apprentice at an innovative design and engineering firm. Erik has worked in mature firms and start-ups, from small to large (Ericsson and ABB). Erik programmed industrial just-in-time (Lean) systems in the 1980’s, was a “process-management guy” in the 1990’s, and spent the fun part of the 2000’s as an agile evangelist-coach. Erik facilitated his first (highly successful) agile project in 2000 and helped Sweden’s top e-commerce brand go agile in 2006. Erik was invited to mentor Ericsson’s first major agile transformation of 2,300 staff at 10 R&D sites in 5 countries 2006-08.

Continuous Delivery is the name of the game both for products and services. Do a feature, deliver it and get feedback as soon as possible. Either split-test or roll-out/roll back. But wait! Why are many organizations failing with – or eroding their software? I feel that there are a few vital pieces are missing: (1) Continuous delivery to me does not mean continuous planning. You need a highly constrained cyclic planning that feeds into iterative development with continuous delivery (learning “just in time”) in order to build true value. (2) Trying to avoid constraints (e.g. #noestimates) is an attempt to avoid making decisions – shying away from innovation and creativity. We need constraints and estimates to generate ideas and make smart choices. Without a constrained system, we will forever chase the electric rabbit of the “unaccountable” customer. (3) Agile teams explore and learn about possibilities well before they need them in the product. Winning teams need to learn what they need “now”.

5  Ken Power

Ken is a Principal Engineer and internal coach and consultant with Cisco Systems. He lives in Galway, Ireland and works with teams and organizations around the world. His responsibilities include leading the agile transformation for Cisco’s largest software group. He also works with universities and research groups in agile, lean and software engineering research. He is currently completing a PhD in Lean Flow and understanding impediments in teams and organizations. He is a frequent speaker at the major international agile, lean and software engineering conferences, and has published numerous papers on agile and lean development. Ken is a Fellow of the Lean Systems Society, a certified Human Systems Dynamics Professional, and a trained Co-Active Coach and Organization & Relationship System Coach.

Continuous Delivery (CD) brings many advantages that enable business agility. However, CD is not solely a technology and infrastructure challenge. Not all teams and organizations are ready to adopt CD. There are a number of pre-requisites that I have found useful to articulate. These pre-requisites are both technical and cultural, and neither can be neglected without jeopardizing the overall effort. Continuous Delivery depends on several pre-requisites being in place. Firstly – the requisite foundation is a culture that supports continuous improvement and problem solving. Secondly – followed by the technical and cultural aspects of Continuous Integration
(CI) and Continuous Feedback. To be successful, these elements must work together to support Continuous Deployment.

6 Linda Rising

Linda Rising is an independent consultant who lives near Nashville, Tennessee. Linda has a Ph.D. from Arizona State University in object-based design metrics. Her background includes university teaching as well as work in industry in telecommunications, avionics, and strategic weapons systems. She is an internationally known presenter on topics related to agile development, patterns, retrospectives, the change process, and the connection between the latest neuroscience and software development. Linda is the author of numerous articles and has published several books: Design Patterns in Communications, The Pattern Almanac 2000, A Patterns Handbook, with co-author Mary Lynn Manns, Fearless Change: Patterns for Introducing New Ideas and to be released in 2015 More Fearless Change.

Even my very elementary understand of physics reveals that light can be viewed as a wave (continuous) or as a stream of particles (discrete). Which is “right”? Both! The useful approach is determined by context. That is so true for models! In our industry, we tend to embrace the current model and throw out everything from the past. We even denigrate past models – not realizing that we are where we are because we built on those former approaches. Perhaps it’s time to become more scientific, to see the effect of context, and to consider appropriate contexts for continuous delivery.

7 Werner Wild

Werner Wild studied Computer Science and Mathematics at the University of Innsbruck and currently teaches at the Free University of Bolzano and the University of Innsbruck. Previous assignments include UNESCO, NIO Goa, ISS The Hague, UBS Switzerland, SwissRe Zurich, Joanneum Research Graz and others. He also helps organizations to build high performance software development teams from scratch, including recruiting, process establishment, project management, training & coaching, dev-ops and also creates scalable architectures. His involvement with computers started 1972; developing virtual machines, compilers, medical and financial applications – and continues with agile trends in Software Engineering. He organizes workshops at international conferences and is an elected official to the Austrian Chamber of Commerce in the Tyrol, identifying and tackling the challenges ahead of the Austrian IT industry. He loves to fly (for more than 25 years) and holds a FAA Commercial Pilot License, with a current Instrument Rating.

In a recent mission critical project (mobile payment) we were able to implement Continuous Delivery successfully. Our goal was to deliver a change within 30 minutes – to production, from code check-in to going live! By fully automating building, testing and deployment (but including a last minute manual safety check!) we were able to bring the time down to 20 minutes! For development we use eXtreme
Programming, steering the project via Kanban, and rigorously following the principles of Lean Software Development. In my opinion a highly disciplined development approach is required to succeed with Continuous Delivery and equally important we should not forget to include our “sys admins” in the process - right from the beginning! We are all in ONE team! However, I am not convinced that the business can leverage this new flexibility to the fullest – since the business has to be as agile as the development team to earn maximum value from Continuous Delivery.

8 Rebecca Wirfs-Brock

Rebecca Wirfs-Brock invented the set of design practices known as Responsibility-Driven Design (RDD) and by accident started the x-DD meme. Along the way she authored two popular object design books. In her spare time she jogs (even in the rain). In her work she helps people hone their design and architecture skills, manage and reduce technical debt, refactor their code, and address architecture risks. She is program director of the Agile Alliance’s Experience Reports Program.

With continuous delivery you can perform small experiments that contribute to small steady course corrections. However, can a more significant innovation fit into a development process where the drumbeat of delivery is constant? Innovation should not be forced to fit into a cycle of continuous delivery. I don’t know how to plan for big innovations but I have some experience with taking slightly innovative ideas off a continuous delivery pipeline in order to vet their feasibility and mitigate design risks before committing to them. We were able to explore unproven, intriguing ideas in small, bounded experiments or innovation spikes. These experiments were several weeks long, and they were time-limited. Some failed. Others succeeded – and through this learning we helped shaped future product innovation.
Learning from Disaster and Experience: Evolving Software Professionalism

Steven Fraser1, Janne Järvinen2, Erik Lundh3, Ken Power4, Linda Rising5, Werner Wild6, and Rebecca Wirfs-Brock7

1 Innoxec, CA, USA
sdfraser@acm.org
2 F-Secure, Finland
janne.jarvinen@f-secure.com
3 Compelcom, Sweden
erik.l.lundh@gmail.com
4 Cisco Systems, Galway, Ireland
ken.power@gmail.com
5 Independent Consultant, TN, USA
risingl@tds.net
6 University of Innsbruck, Austria
werner.wild@uibk.ac.at
7 Wirfs-Brock Associates, OR, USA
rebecca@wirfs-brock.com

Abstract. Professionalism evolves as knowledge and skills mature from craft to commercial practice – often as the result of learnings derived from failure and human hazard. Aviation, medicine, engineering, and architecture are examples of disciplines with an established knowledge base and curriculum of learning and mentorship. These disciplines often require regulated practices executed by certified professionals to ensure the safety and economic value of delivered services. This panel will debate whether we are learning effectively from our experiences and what might be done to accelerate increased software professionalism and product value.

Keywords: Professionalism · Knowledge · Skills · Craft · Practice · Learning · Failure · Certification · Value · safety

1 Steven Fraser (Panel Impresario)

Steven Fraser is based in Silicon Valley California and has served as an innovation catalyst with global influence for three Fortune 500 Companies serving as: Director, Cisco Research Center; Senior Staff, Qualcomm Learning Center; and Senior Manager, Nortel Disruptive Technologies and Global External Research. In addition to a year spent as a Visiting Scientist at Carnegie Mellon University’s Software Engineering Institute (SEI) consulting on Domain Engineering (software reuse) processes and practices he has organized and delivered over 75 software engineering conferences, panels, keynotes, workshops, and tutorials.
Learning from Disaster and Experience: Evolving Software Professionalism

Twenty-five years ago, CMU professor Mary Shaw wrote on “Prospects for an Engineering Discipline of Software” in IEEE Software (Nov.1990, pp 15-24). She proposed that engineering disciplines evolve from craft – with development characterized by virtuosos, talented amateurs, intuition, brute force, extravagant use of available materials and manufacture for use rather than sale. Shaw further postulated that as time progresses, a shared experience base characterized by skilled crafts people, established procedures based in science (enabling education and training) with a concern for economics and materials – scales to commercial practice and manufacture for sale. “Engineering discipline” is achieved when a cadre of educated professionals with a shared knowledge in analysis, theory, and science – including awareness for economics, ethics, human hazards and risks – evolves to develop products and services for the good of humanity. This panel brings together industry experts to share their thoughts on software professionalism and discipline.

2 Janne Järvinen

Janne Järvinen has over 25 years of experience in software engineering and software process improvement in various positions within software industry from programmer to VP Engineering in small and large software companies. Janne is Director, External R&D Collaboration at F-Secure Corporation and has also been active in academia in various research programs such as European ESPRIT (BOOTSTRAP, PROFES) and ITEA (MOOSE, FLEXI). He led the Cloud Software program (www.cloudsoftwareprogram.org), and now leads the N4S program (www.n4s.fi) under DIGILE Oy. He is an IEEE member and holds a PhD in Information processing science from University of Oulu (2000).

I am a strong believer of continuous improvement in all possible forms. I have led software teams both in small and large organizations and I have seen both successes and failures. More than often an important contributing factor to success has been the ability learn from one’s mistakes. On the other hand, as making of software is largely invisible software professionals are often challenged to demonstrate aspects of their work that may bear little relevance to actual software. Or what actually is relevant for a software professional? Is it enough that one produces superior code? Finally, I would borrow ideas to develop software professionalism from digital security research where I have seen teams of analysts engage in investigations of the latest threats in close cooperation with industry partners and international and national information security authorities. We must share our experiences with a larger community more effectively and transparently than possible today.

3 Erik Lundh

Erik Lundh has more than 30 years’ experience in product-technical-software development beginning as an apprentice at an innovative design and engineering firm. Erik has worked in mature firms and start-ups, from small to large (Ericsson and ABB). Erik programmed industrial just-in-time (Lean) systems in the 1980’s, was a
“process-management guy” in the 1990’s, and spent the fun part of the 2000’s as an agile evangelist-coach. Erik facilitated his first (highly successful) agile project in 2000 and helped Sweden’s top e-commerce brand go agile in 2006. Erik was invited to mentor Ericsson’s first major agile transformation of 2,300 staff at 10 R&D sites in 5 countries 2006-08.

We lost the ball when the Agile movement was handed “sticks”: the 1990s lightweight method Scrum – reengineered as the safe haven for change-angst mediocre low level managers (in my opinion head-count reduction road-kill), Kanban – the Taylorists revenge, and SAFE – the RUP exploiters revenge. Good people turned away in disgust from third rate agile “profiles”, catering to pointy-haired bosses with money in hand looking for agile storefronts on assembly line software sweatshops. Some of the best of the original agile proponents stopped caring about process and product. Their reaction was to initiate the blue collar Software Craftsmanship Movement. In my opinion, we need agile software professionals that master process, product, craft and technologies – and we need to send all scrum masters to retraining as product management support. I spent the 1980s solving – strike that – *fixing* larger and larger problems with my technical prowess, only to realize that the root causes were non-technical. Early agile methods involved programmers in business and vice versa, and that as what attracted me and my Swedish process improvement community to agile methods like Extreme Programming in 1999-2000. That hope and motivation has been replaced by nervous laughs when asked whether agile works.

4 Ken Power

Ken is a Principal Engineer and internal coach and consultant with Cisco Systems. He lives in Galway, Ireland and works with teams and organizations around the world. His responsibilities include leading the agile transformation for Cisco’s largest software group. He also works with universities and research groups in agile, lean and software engineering research. He is currently completing a PhD in Lean Flow and understanding impediments in teams and organizations. He is a frequent speaker at the major international agile, lean and software engineering conferences, and has published numerous papers on agile and lean development. Ken is a Fellow of the Lean Systems Society, a certified Human Systems Dynamics Professional, and a trained Co-Active Coach and Organization & Relationship System Coach.

The profession of software engineering is still finding its identity, even now decades on from when the term was first coined. Referring to professional education in general, Donald Schön describes what he calls “the crisis of confidence in professional knowledge”. Schön proposes that university-based professional schools should take influence from the traditions of education for practice. These influences should include art and design, music and dance conservatories, athletics coaching, and apprenticeship in the crafts. All of these have in common an emphasis on coaching and on learning by doing. He goes even further to say that professional education needs to be redesigned “to combine the teaching of applied science with coaching in the artistry of reflection-in-action”. There are several options for bringing this to life in teams.
and organizations. Examples include job rotations, shadowing, and formal mentorship. The increasingly popular guilds, chapter and communities of practice are further opportunities, but they need to have real substance and organizational support.

5  Linda Rising

Linda Rising is an independent consultant who lives near Nashville, Tennessee. Linda has a Ph.D. from Arizona State University in object-based design metrics. Her background includes university teaching as well as work in industry in telecommunications, avionics, and strategic weapons systems. She is an internationally known presenter on topics related to agile development, patterns, retrospectives, the change process, and the connection between the latest neuroscience and software development. Linda is the author of numerous articles and has published several books: *Design Patterns in Communications, The Pattern Almanac 2000, A Patterns Handbook*, with co-author Mary Lynn Manns, *Fearless Change: Patterns for Introducing New Ideas* and to be released in 2015 *More Fearless Change*.

This topic calls to mind the ACM Forum on Risks to the Public in Computers and Related Systems moderated by Peter G. Neumann ([www.risks.org](http://www.risks.org)) detailing software misadventures. The Forum reports accidents, injuries, and deaths as a result of software error. I always read that and I always thought that our industry didn't seem as responsive as other fields such as avionics. We don't seem to pay attention to those reports. We seem to talk about the latest buzzword and I wonder what we are carrying forward. Are we as an industry becoming like our projects – small and focused on the moment – just trying stuff without any formal approaches and leaving our mathematical and scientific history behind? As software becomes a more important part of every produce and service. This failure to learn seems to lead to a place we might not want to go.

6  Werner Wild

Werner Wild studied Computer Science and Mathematics at the University of Innsbruck and currently teaches at the Free University of Bolzano and the University of Innsbruck. Previous assignments include UNESCO, NIO Goa, ISS The Hague, UBS Switzerland, SwissRe Zurich, Joanneum Research Graz and others. He also helps organizations to build high performance software development teams from scratch, including recruiting, process establishment, project management, training & coaching, dev-ops and also creates scalable architectures. His involvement with computers started 1972; developing virtual machines, compilers, medical and financial applications – and continues with agile trends in Software Engineering. He organizes workshops at international conferences and is an elected official to the Austrian Chamber of Commerce in the Tyrol, identifying and tackling the challenges ahead of the Austrian IT industry. He loves to fly (for more than 25 years) and holds a FAA Commercial Pilot License, with a current Instrument Rating.
In aviation many systems are in place to learn from experience and to receive valuable feedback. For example, we must continually demonstrate our skills in a simulator and in flight. If there are “incidents” – these must be reported (e.g. ASRS: Aviation Safety Reporting System - http://asrs.arc.nasa.gov). Pilots frequently review incident and accident reports to learn from others. Perhaps it is time to borrow an idea or two from aviation to foster increased professionalism in our software industry.

7 Rebecca Wirfs-Brock

Rebecca Wirfs-Brock invented the set of design practices known as Responsibility-Driven Design (RDD) and by accident started the x-DD meme. Along the way she authored two popular object design books. In her spare time she jogs (even in the rain). In her work she helps people hone their design and architecture skills, manage and reduce technical debt, refactor their code, and address architecture risks. She is program director of the Agile Alliance’s Experience Reports Program and co-chair of the XP 2015 Experience Reports track.

I hope I never stop learning. My technical learning has been driven by positive experiences and interactions with others as we build something together. I learn the most when I work with those who are articulate and considerate enough to explain why they want to solve something some particular way, to share what the next move we should take as well as why they are confident (or not) about what we’re doing, and how they feel about our code. Back and forth communication as we build something significant and complex together is where I learn best. A highly performing, tight agile team is one of the best opportunities for professional learning. However, I have written lots of code solo, and through those experiences I have also learned a great deal. I don’t believe that every bit of code is best developed by a collective mind. Sometimes solo efforts are the most productive where you get deep into the problem.

I have also worked on teams where we didn’t communicate as we banged out code and as a consequence, we didn’t learn much. What seems most important to me as a software professional is doing things that help me develop a wise inner critic who considers why I prefer some solution over another, why I think this is a hack and that is not, why I think this will work and that will not. And that is because software is, well, soft. There are many ways to solve a problem in code, but only a few are good enough solutions. To get really good at this profession, you need challenging work where you don’t know how to solve it and lots of feedback.
Practical Applications of the Agile Fluency Model

Diana Larsen¹, Steve Holyer², Jutta Eckstein³, Antti Kirjavainen⁴, and Olli Sorje⁵

¹ FutureWorks Consulting LLC, 6307 NE 8th Avenue, Portland, OR, USA
Diana@futureworksconsulting.com
² Engage Results, Founder, Agile Coaching, Aemtlerstrasse 114, Zurich, Switzerland
coach@engage-results.com
³ IT communication, Gaussstr. 29,38106 Braunschweig, Germany
je@it-communication.com
⁴ Flowa, Department, Kaarlenkatu 3-5 B 14, FI-00530 Helsinki, Finland
Antti.Kirjavainen@flowa.fi
⁵ Affecto Finland Oy, Retail and Services, Helsinginkatu 15, FI-20500 Turku, Finland
olli.sorje@affecto.com

Abstract. First published in 2011, the Agile Fluency Model (agilefluency.com) describes four stages for achieving agile team proficiency: focus on value, deliver value, optimize value, and optimize the system. Adopting agile in successive stages the model delivers benefits proportional to investments. The model fosters the identification, adaption and adoption of agile practices appropriate to the team’s context and objectives. Does it reflect reality? Is it useful? Does it work when theory meets practice? This panel brings together Agile Fluency practitioners to share their challenges, successes, and future directions for model evolution.

Keywords: Agile adoption · Agile fluency · Agile practices · Team proficiency · Value delivery

1 Diana Larsen – Panelmeister

Diana consults with leaders and teams to create work processes and environments where innovation, inspiration, and imagination flourish. She is an international authority in the areas of Agile software development, team leadership, and Agile transitions.

Diana co-authored Agile Retrospectives: Making Good Teams Great! and Liftoff: Launching Agile Teams and Projects. She created the Agile Fluency™ Model with James Shore.

2 Steve Holyer — Looking for Organizational Fluency

When Diana showed the Agile Fluency™ Model to me, I recognized that it describes exactly what I’ve experienced when I’ve worked with software teams who are successfully mastering Agile development. That’s why I’m so excited about it. It’s not a
maturity model that dictates how teams must progress, it’s an observation based on experience of what teams can expect in their progress. I’ve used the model as a tool to encourage teams that were stuck doing “by the book Scrum” (1-Star) so they could see a much richer path. The Agile Fluency™ Model has proven useful for changing the conversation with leaders about targets and KPI’s so that it becomes a conversation about helping teams and the organization evolve their understanding of leadership and team learning. Diana and I used the Agile Fluency™ Model to create a set of learning objectives that we’ve used to train teams at each level of Agile Fluency. This broadened the focus of our training beyond the typical Agile training focus which is often focused on just one level of fluency.

The Agile Fluency™ Model is written to describe the evolution of a development team. Therefore it naturally describes the experience of delivering software from the perspective of the builders creating code. And yet, the model itself states there’s a shift in organizational structure that takes place as a build team becomes a 3-star team. There’s also an organizational culture shift that takes place as the build team moves from a 3-star team to become a 4-star team. This begs the question, “How does the whole organization become fluent at the 3-star and the 4-star level?” There must be more to the Agile Fluency™ Model that hasn’t been described yet at the level of Agile Management. The Agile Fluency™ Model is based on anecdotal evidence from the authors and their colleagues. Now that the model is defined, I’m looking forward to expanding the body of knowledge about Agile Fluency with efforts to collect more data so we can see what new data reveals.

Steve serves as advocate, trainer and mentor for companies discovering different ways of working using Agile practices in a productive, fulfilling, and fun way. He uses the model in most, if not all, aspects of his work with teams.

3 Jutta Eckstein – The Shifts are Harder than You Think

To me, as a practitioner, the Agile Fluency™ Model mirrors what I see happening in Industry. I experience that many teams already struggle with the first stage on the model – to focus on value. They concentrate instead on the adherence to practices. Yet, I also experienced teams that not only focused on value but reached the second stage to also deliver it. They were not able to let go of their habits even when asked to switch back to waterfall. They just couldn’t.

However, the model doesn’t show how hard it is to move from one stage to the other. And I wonder if reaching four stars is only possible if an organization starts this way. Although I want to believe it’s possible, I doubt that a cooperation doing business in a non-agile way for e.g. 30 years will be able to ever reach that stage. If only for the reason, that they won’t have the patience to go through the deep culture shift which will take at least ten years.

Jutta Eckstein works as an independent coach, consultant, and trainer. She holds a M.A. in Business Coaching & Change Management, a Dipl. Eng. in Product-Engineering, and a B.A. in Education. She has helped many teams and organizations worldwide to make an Agile transition. She has a unique experience in applying Agile processes within medium-sized to large distributed mission-critical projects. She has published her experience in her books *Agile Software Development in the Large,*
Agile Software Development with Distributed Teams, Retrospectives for Organizational Change, and together with Johanna Rothman Diving for Hidden Treasures: Finding the Real Value in your Project Portfolio.

4 Antti Kirjavainen

Having coached software development and knowledge work teams and organizations for the past 5 years, I have found that setting goals together with the organization I’m coaching helps to carry out the coaching relationship and contributes to the relationship’s success. We set the goals in terms of impact. I have used the Agile Fluency™ Model over the past 2.5 years to facilitate this goal-setting and to successfully set expectations related to the change process.

I have found Agile Fluency™ Model useful for agreeing on the current state of teams with the customer, agreeing on the desired end state with teams and their sponsors and with setting realistic expectations for the change. What I have found challenging is making the progress visible and measurable during the change. The metrics described in the model work well as lagging indicators, but we would benefit from leading indicators as well. The linear representation of the model makes it easy for sponsors to mistake it for a maturity model. There is some ambiguity in the description of the model and its stages and whether or not to focus on different stages at once. That is why I have adapted my own description of model’s stages to call them dimensions instead.

Antti is an entrepreneur at Flowa (Finland). He helps organizations achieve effectiveness and excellence in software development and knowledge work through agile, lean thinking and Management 3.0. Antti is also fascinated about the potential of games for learning and facilitating collaboration.

5 Olli Sorje

For me the Agile Fluency™ Model means giving context for all Agile practices. It helps you to identify where you and your team currently are and decide where you should aim for. Not all organizations should aim for a 4-star level. This model also helps you understand why you are not maybe getting the benefits that Agile promises. I also like the concept of Fluency a lot. You might use all the practices from a specific star level, but if you devolve back to old habits under pressure it means you aren’t fluent at that level! That’s unfortunately what typically happens in Agile transitions. You adopt Agile practices, but then the deadline starts to get closer. You decide that these new practices are slowing you down, and you fall back to old practices.

Olli Sorje is working as Lead Developer at Affecto Finland. Olli is passionate about Agile and creating real value for customers. Olli enjoys working with teams and helping them evolve.

References

Doctoral Symposium Abstracts
Improving Processes by Integrating Agile Practices

Philipp Diebold
Fraunhofer IESE, Kaiserslautern, Germany
philipp.diebold@iese.fraunhofer.de

Abstract. Even though agile development has been known for many years, it is mainly used in information systems and is not common yet in embedded systems. Despite the mandatory regulations, the companies would like to increase the flexibility by the benefits of agile development. Thus, the idea is creating an Agile Capability Analysis with a subsequent Process Simulation, resulting in appropriate process extensions adhere to mandatory requirements. Because often agile methods require context-specific adaptations, we believe that the up-front investigation which agile practices to integrate into processes entails many benefits, especially in regulated domains.

Keywords: SPI · Agile SW development · Agile practices

1 Introduction and Problem

Agile development has been common for several years. Nonetheless, it is mainly used in information systems and not in embedded systems (ES), which are often restricted by regulations. The problem in these domains is their inflexible development. Thus, they want to use agile development within their regulations. This raises the research question: How to bring more agility to the regulated software domains?

2 Related Work

Even though plan-based processes, e.g. waterfall and V-model, which fit many regulations, dominate different ES domains [2], there are some approaches that try to address the problem of getting agility integrated into regulated domains. Some of the common agile methods like Scrum are already used in part in ES. But this only applies to pilot projects or to early phases of the final product, where it is possible to try things and where developers are not restricted by too many regulatory requirements. In addition, less specific approaches have also been published that try to address the issue of more flexibility in regulated domains. One is the Agile V-Model [1], developed for medical devices.

As shown in the short discussion of existing related work and the already mentioned issue of adapting the different agile methods, there is a lack of systematic Software Process Improvement (SPI) approaches in research that make use of agile practices while adhering to mandatory regulations.

© Springer International Publishing Switzerland 2015
DOI: 10.1007/978-3-319-18612-2
3 Solution Approach

To address this problem, the overall solution idea is based on two parts: the Agile Capability Analysis and a Process Simulation aimed at finding the most appropriate agile practices as process extension. These two parts constitute the contribution and novelty of the idea because no such analysis is currently available and even though process simulations are widely used, none deals with the integration of agile practices.

The purpose of the Agile Capability Analysis is to identify the capability of extending the current development process with agile practices in terms of the context (incl. regulations). The inputs are the current process and context information, e.g., team size or team location. This analysis is built on a model containing the information about the impact of all different agile practices, which are formal described in a repository. This reveals all possible agile practices that could be potential extensions in the specific context.

The purpose of the Process Simulation is to simulate possible agile practices combinations and their joined impact. Therefore, the formal process description and the outputs of the analysis are used. All possible combinations of the set of agile practices are simulated in the company’s process. Thus, the formalized process and the possible process extension with the necessary information, e.g. their impact, will be used to come up with the simulation model. This model will then be used for the simulation and should provide information on how the combinations might behave concerning the impact characteristics.

Based on this information, the SPI can be performed by selecting one of the possible combinations that best fits to the organizations improvement goal. This should than be implemented and further been evaluated compared to the assumed impact.

4 Conclusion and Future Work

The overall solution idea should support SPI in regulated environments by integrating agile practices into the current development processes. This is done by the Agile Capability Analysis and a simulation of the process extensions, the agile practices.

The future research agenda regarding this idea includes the following aspects: The Agile Capability Analysis will be defined in detail because only parts already exist. Additionally, the Process Simulation needs to be elaborated more detailed because currently it is rather a high level idea. Finally, the question is how to evaluate the overall idea and/or its different parts.

References

Assurance Case Integration with An Agile Development Method

Osama Doss and Tim Kelly

High Integrity Systems Engineering Research Group,
Department of Computer Science, University of York, York, YO10 5DD, UK
{osad500,tim.kelly}@york.ac.uk

Abstract. Agile software development has had success in different domains. However there is one area where the implementation of agile methods still needs development – that is in the field of safety critical systems. In this field, the software engineering processes need to be justified against the requirements of software safety assurance standards (such as ISO 26262 in the automotive domain). We describe our ongoing research on assurance case integration with an existing agile development method – SafeScrum.

Keywords: Safety-critical systems · Assurance · SafeScrum · Assurance case · Safety cases

1 Research Problem

Whilst the use of agile methods is seen by some as attractive, and there is some evidence of increasing use in safety-critical domain, there are still many in the domain who have concerns. For example, Redmill [1] raises concerns about whether hazard identification and analysis can be carried out incrementally. The question of how to integrate agile methods and safety assurance is not new. But there is one particular area of practice that remains neglected in the existing work – namely the integration of safety (assurance) case development with an agile approach. A safety case is the argument and evidence that establishes the acceptable safety of safety-critical system [2]. It is normally prepared (by the developer) and assessed (by an independent assessor or regulator) as part of safety critical systems development. Safety cases are an increasingly widespread approach [3]. Structured argumentation approaches (such as the use of the Goal Structuring Notation – GSN - [4]) have become popular as a means of explicitly representing the arguments (and links to evidence) contained within a safety case. The research problem we are tackling is the integration of assurance case development (including the incremental development of structured arguments) with a typical agile development method.
2 Research Methods

- We have conducted a survey to investigate the practical problems posed by the integration of the two disciplines.
- We are developing a pattern-based approach to integrating software safety cases, SafeScrum’s Safety Product Backlog, risk-based planning, and requirements-based evaluation. Software safety argument patterns describe the nature of the argument and safety claims that would be expected for any software safety case [5].
- The feasibility, and practicality of the proposed integration of safety case development with SafeScrum will be initially evaluated through an illustrative case study.
- Peer review (through structured questionnaire) of the developed approach applied to the worked case study example will be conducted.
- 1-to-1 semi-structured interviews will also be used with some of the respondents from our initial survey, the purpose of this interview study is to investigate the success proposed of safety case development within SafeScrum.

3 Results and Future Work

We have already conducted the practitioner survey (with 31 respondents). The results from this survey have provided a clear direction in terms of the importance of incremental hazard analysis, safety requirements development, and assurance case development (i.e. they indicate clearly that these activities must be performed within an increment, rather than simply being up-front or end-of-development activities). We are using the insight gained from the case study to evaluate how safety activities are currently being proposed within the Safe-Scrum method, and to help define a process model for how requirements development, hazard analysis and assurance case development can be performed as in-increment activities. With regard to assurance case development, we have identified that the existing GSN argument patterns of Hawkins et al. [5] already attempt to integrate software safety requirements development and assurance case development. At present these patterns are expressed to suit a traditional ‘tiered’ software development. We are currently examining how these patterns can be adapted to suit incremental development. Following development of the process model, and adaptation of the patterns we will be applying the proposed approach to a case study system to serve as the basis for further evaluation with the respondents from our initial survey.

References

Data-Driven Decision-Making in Product R&D

Aleksander Fabijan¹, Helena Holmström Olsson¹, and Jan Bosch²

¹ Malmö University, Faculty of Technology and Society, Östra Varvsgatan 11, 205 06 Malmö, Sweden
{Aleksander.Fabijan, Helena.Holmstrom.Olsson}@mah.se
² Chalmers University of Technology, Department of Computer Science and Engineering, Hörselgången 11, 412 96 Göteborg, Sweden
Jan.Bosch@chalmers.se

Abstract. Software development companies experience the road mapping and requirements ranking process to be complex as product management (PdM) strives in getting timely and accurate feedback from the customers. Often, companies have insufficient knowledge about how their products are being used, what features the customers appreciate and which ones will generate revenue. To address this problem, this research aims at helping the companies in closing the ‘open’ feedback loop that exists between PdM and customers. Moreover, the research strives at exploring techniques that can be used to involve customers in continuous validation of software functionality in order to provide PdM with the evidence needed for accurate R&D investments.

Keywords: Customer feedback · Data collection · The ‘open loop’ problem

1 Introduction

Due to the increasing amount of software in products and hence, the capability to connect these products to the Internet, there is a fundamental shift in how products are developed and how the life cycles of these products are perceived [1], [2]. This implies that user feedback collected in the early phases of product development is complemented with another data source, i.e. product data revealing real-time use [3].

Despite this, product managers often struggle with getting timely and accurate feedback from customers. Typically, the feedback loops are slow and there is a lack of mechanisms that allow for efficient customer data collection and analysis [1], [5]. As a result, companies have insufficient knowledge about how their products are used and what features the customers actually appreciate. This means that there is an ‘open loop’ between customer data and product management decisions [4], [5].

To address this problem, this thesis project aims at exploring the existing methods and discovering new techniques that allow for (1) continuous validation of deployed software functionality through product and customer data, and (2) efficient customer and product data collection and analysis practices.
2 Proposed Approach and Evaluation of Results

The research presented in this proposal is conducted within a large research collaboration consisting of three universities and eight companies.

For the purpose of this research, we use a qualitative case study approach in which we engage with company representatives with various roles such as e.g. developers, product managers and product owners etc. on a continuous basis using a mix of interviews, workshops, weekly status update meetings, individual visits and validation sessions. Interview results work as a basis for workshop sessions and seminars in which we further discuss and validate our findings. It should be noted that the company collaboration is well established and results are continuously evaluated with the companies and reported every six months.

In terms of data analysis, we adopt a qualitative approach as described by e.g. Walsham [6].

3 Expected Contributions and Progress Towards the Goals

As the first phase in the research project outlined here, we have conducted a ‘state-of-the-art’ literature review in which we identify existing customer feedback and data collection techniques as reported in the software engineering domain, i.e. the top ranked journals and conferences [7]. The results were summarized in a structured model that provides an overall understanding for the existing feedback and data collection techniques, and that works as a support for selecting the appropriate feedback technique(s) in a specific stage of the software development process.

Moving forward, we plan to expand this model with additional feedback collection techniques, identify and complement it with the types of data collected, and to validate our model in the companies to provide also a ‘state-of-practice’ view.

References

Combining Kanban and FOSS: Can It Work?

Annemarie Harzl and Wolfgang Slany

Institute of Software Technology, Graz University of Technology,
Inffeldgasse 16b/II, 8010 Graz, Austria
aharzl@ist.tugraz.at, wolfgang.slany@tugraz.at

Abstract. Free and Open Source Software (FOSS) and Agile Software Development (ASD) research have gained momentum over the past decade. However, to the best of our knowledge, there exists no work about these two phenomena combined. This thesis will show how Agile Software Development (ASD), specifically the Kanban Method, and FOSS can be consolidated and how they can benefit from each other’s advantages. The agile community and the FOSS community can benefit from this body of work, as we aim at broadening the understanding of both.

Keywords: Lean · Agile · Kanban · FOSS · Open source

1 Statement of Research Problem

In 2009 Ågerfalk et al. [1] identified research about agile development in the context of open source software as a future research area. This research intends to take first steps to fill this gap. The knowledge if and how agile methods should be introduced into FOSS projects can be valuable for the FOSS community and cooperating companies, which intend to spread their agile methods to the community, they are working with.

For the purpose of this work a FOSS project was selected, which is situated at Graz University of Technology. One team was chosen to participate in the study, because conducting the study with the whole project, would not be feasible. Being at the same university as the FOSS project provides us with the opportunity to directly observe the contributors in their natural context. The project already applies some agile methods (for example Test-Driven Development (TDD)), but not in a systematic way, and contributors experience some problems with agile adoption. Hence, they contacted us and asked how they could improve their use of agile methods. The Kanban Method [2] has been chosen by the research team, because it is the most adaptive method [3]. It allows to take an existing process and slowly adapt it to the specific needs of the organization. The Kanban Method does not impose a huge set of rigid rules, which is very important, because introducing rigid rules into a system almost always requires positional power, which is not available in a FOSS community.
Combining Kanban and FOSS: Can It Work?

Research Questions: My doctoral research aims at answering the questions if Agile Software Development (ASD) techniques, specifically the Kanban Method, can be successfully applied to a FOSS project and whether a FOSS project can benefit from it. I am interested in the current state of agile adoption in the studied project and how the agile process and its introduction into the project can be customized to better fit the needs of the FOSS project and its contributors.

- Research Question (RQ)1 What are the current problems with agile adoption and can they be solved through the use of the Kanban Method?
- RQ2 Can FOSS projects benefit from using agile methods like the Kanban Method?
  - RQ2.1 Do FOSS contributors, who are coached in the Kanban Method, experience this knowledge as beneficial to their work or not?
  - RQ2.2 Do interaction or communication in the team change with the use of the Kanban Method?
  - RQ2.3 Does the Kanban Method have any effect on the source code created by the coached team?

2 Research Methodology

To answer RQ1 a longitudinal single-case study was chosen as research methodology because the case study method is suitable for research of contemporary phenomena in their natural context [4]. According to [5] this method has advantages when a researcher has little or no control over the studied events and when, how or why research questions are asked. At first I will observe team meetings and coding sessions and then I will coach the team in the usage of agile methods. To answer RQ2.1 another research method will be used, namely surveys. Participants of the case study will give subjective feedback about their experience with agile methods at the beginning, throughout and at the end of the case study. They will do this through questionnaires. Surveys and observations of team meetings will be used to answer RQ2.2. This provides the opportunity to compare the perception of team members and someone outside the team. Some code metrics will be calculated for the teams’ source code, for example code complexity as a measure for internal quality, to answer RQ2.3.

No results have been achieved so far.

References

Paradigm Shift from Large Releases to Continuous Deployment of Software: Designing a Reference Model for Continuous Deployment

Teemu Karvonen, Markku Oivo, and Pasi Kuvaja

University of Oulu, Finland
{Teemu.3.Karvonen,Markku.Oivo,Pasi.Kuvaja}@oulu.fi

Abstract. Continuous deployment (CD) is an essential method as software development companies move towards real-time business and continuous experiments. Powered by the lean and agile methods, CD aims for continuous deployment of valuable software. This doctoral research investigates what it will take to enable CD. The findings will be collected to generate a CD reference model. The research is initiated by studying existing literature and models for organisational assessment in relation to lean and agile approaches. Next, the focus is sharpened to capabilities that are required for enabling CD in information and communication technologies (ICT) industry. The research will apply literature reviews, case studies and the design science research (DSR) framework.

Keywords: Lean · Agile · Software development · Continuous deployment · Real-time business · Reference model · Assessment

1 Doctoral Research Plan

The software engineering (SE) field of research has focussed on studying multidisciplinary aspects concerning how to improve ways of developing and deploying software intensive products. Today, the development paradigm in information and communication technologies (ICT) has moved from plan-based waterfall to iterative agile methods [1]. Many software development companies have set their future goal to ‘deliver value in real time’. It seems that innovation experiment systems and continuous deployment of valuable software are essential new capabilities which companies must develop in order to advance beyond the agile approach [2,3,4]. Continuous deployment (CD) [5,6] describes practices for developing and deploying software. CD has clearly built-in principles of lean [7] and agile. According to the lean principle ‘flow’, value-creating steps should occur in a tight sequence so that the value will flow smoothly towards the customer. In addition, the agile manifesto\(^1\) describes how to deliver software: ‘Our highest priority is to satisfy the customer through early and continuous delivery of valuable software’. Moreover, it aims to ‘deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale’.

\(^1\) http://agilemanifesto.org/principles.html

© Springer International Publishing Switzerland 2015
DOI: 10.1007/978-3-319-18612-2
Organisational assessment models are also frequently discussed in SE studies. Periodical assessments are found useful for analysing an organisation’s current status. A reference model can be used together with an assessment model to define desired real-life working practices. The reference model (what to assess) and assessment model (how to conduct assessment) can also be integrated into a single ‘tool’, such as in Lean Enterprise Self-Assessment Tool (LESAT) [7].

The research goal is to develop new knowledge and methods which aid in the transition towards CD. The main research methods and questions are as follows: 1) How do ICT companies apply CD? – Case studies; and 2) How can CD capabilities be assessed? – Systematic mapping study and design science research (DSR) [8]. In the principles of DSR, the aim is to produce innovative research artefacts. Research has already started, with a wide-range study of lean and agile methods and their related assessment models. The first design artefact, LESAT for software [7], is an adaptation of the self-assessment tool designed for lean enterprise transformation. Next, the research scope will be narrowed to CD practices and capabilities in a real environment, following the design and validation of the CD reference model. The first results from case study interviews indicate that CD requires the involvement of multiple stakeholders and the tight integration of the customer, product management, research and development (R&D) and operations.

References

How to Adopt Continuous Delivery? A Research Proposal

Eero Laukkanen¹ and Casper Lassenius¹,²

¹ Aalto University, P.O. BOX 19210, FI-00076, Aalto, Finland
{eero.laukkanen,casper.lassenius}@aalto.fi
² Massachusetts Institute of Technology, Sloan School of Management

1 Research Problem

Continuous delivery (CD) is a software development discipline in which software can be released to production at any time [1]. The discipline is achieved through optimization, automatization and utilization of the build, deploy, test and release process [2]. The proposed benefits of CD are increased visibility, faster feedback and empowerment of stakeholders [2]. However, when trying to adopt CD, organizations have faced numerous challenges and problems [3, 4]. Even continuous integration (CI), which is a prerequisite for CD [2], has not been adopted in some cases [5].

The dissertation aims to gain deep understanding of the problems when adopting CD and build solutions to those problems. Since the already found problems have been numerous [3, 4], we believe that simply finding the problems is not enough, but we need to analyze the causal structure of the problems in order to solve them. Thus, the following research questions are asked:

1. What problems are faced when adopting CD for a software product?
2. What are the causes for those problems?
3. What solutions can be used for solving those problems?

2 Research Methodology

The dissertation will consist of five empirical qualitative studies. First, a systematic literature review (SLR) [6] is executed to summarize existing research and experience reports on the subject. The SLR will include empirical studies and experience reports which are qualitatively synthesized into a theory of adoption problems and solutions. Second, the theory is refined through a single case study [7] focusing on problems perceived by individuals in a project adopting CD. Individual analysis is done, because it is believed that the perceptions between individuals differ. Third, a single case study will be performed using ARCA root cause analysis method [8] to study adoption problems in a single case and to innovate solutions to the problems. Fourth, the impact of these solutions is investigated in a follow-up study. Fifth, the root cause analysis is
performed in other cases too, and the results are synthesized into a generalized theory.

Data sources for the case studies are Finnish software companies in the Digile Need for Speed program that continues until 2017. Thus, there is an opportunity to conduct long-term longitudinal case studies of CD adoption. The selection of data sources for the case studies is done partly based on convenience, but also according to the research goals. The subject of the research program is tightly related to CD, so it is expected that the case companies are also valid subjects for the studies.

3 Results and Future Agenda

The systematic literature review is in writing and will complete in near future. Data collection for the single case study has been executed partially. The design for the first root cause analysis case study has been done and the data collection will begin in the near future.

The initial results of systematic literature review indicate that numerous problems when adopting CD have been identified and some of them could be solved. However, we found that there is a lack of causal analysis done on the problems. This indicates that our future agenda on root cause analysis will provide improvement on the current knowledge.

References

Posters
Teaching Scrum – What We Did, What We Will Do and What Impedes Us

Emil Alégroth, Håkan Burden, Morgan Ericsson, Imed Hammouda, Eric Knauss, and Jan-Philipp Steghöfer

Abstract. This paper analyses the way we teach Scrum. We reflect on our intended learning outcomes, which challenges we find in teaching Scrum and which lessons we have learned during the last four years. We also give an outlook on the way we want to introduce and apply Scrum in our teaching and how we intend to improve the curriculum.

The results aggregated in this paper are a response to a crisis: we, a group of teachers and researchers at the University of Gothenburg (GU) and Chalmers Technical University have realized that the way we teach Scrum doesn’t align with the intended learning outcomes of our courses. These often include process knowledge, technical knowledge, and methodological knowledge. The intended outcome in the process area is the ability to apply the process and the agile principles in future development projects. We feel that the courses become unmanageable for us and put a lot of cognitive load on our students who learn Scrum in lectures and apply it in projects in parallel. While applying Scrum is key to understand it, the projects’ deliverables and technical aspects tend to shadow teaching goals and reflection on agile principles and practices. This causes students to feel like the process is “overhead” instead of being a possibility to structure their work. Since we as teachers are necessarily mainly concerned about the outcome of the project and mainly interact with the students when they present deliverables, we also have very little opportunity to observe the application of the process and give direct feedback on it.

The Software Engineering division is a joint venture between GU and Chalmers. Apart from courses in different programmes at both universities, it offers a Bachelor in Software Engineering and Management and a Master in Software Engineering. Both programmes emphasise project-based learning and thus allow students to experience work in group settings with complex case studies and fixed deadlines. A number of these courses either include agile practices in their learning objectives or make use of them for the project work.

As teachers at the Software Engineering division we are responsible for teaching Scrum in four courses in three contexts – Software Processes (first term) and Software Architecture Project (third term) in the Software Engineering and
Management bachelor program; Software Engineering Project (second term) for various engineering and IT programs as well as Agile Software Development in the Software Engineering master program. The project courses in the bachelor program and the course in the master program have a focus on technical knowledge whereas the Software Processes course focuses on theoretical knowledge.

By sharing our experiences and our lessons learnt we hope to allow trainers and teachers elsewhere to benefit from our experience and to balance the cognitive load and to start the same alignment process in education and training of agile software development that has now begun for us.

To get an overview of how we teach and apply Scrum in our courses we collectively reflected on our experiences and, applying the terminology of Brookfield, evaluated our practices using a peer lens [1]. Following Schön the reflection was on-action [2] since our teaching is spread out over the academic year and the course curricula are set for each course instance. This process led us to a number of insights and best practices that we want to apply in our future teaching. Two examples are:

1. Teaching Scrum needs to have a practical element: Scrum must be applied to know it. However, the more technically advanced the practical element is, the more the students focus on learning the platform instead of the process. The platform becomes an impediment for learning Scrum.

2. Stress is an impediment for learning Scrum since the students focus on delivering before the deadline instead of using a sound process. In a workshop setting the stress creates learning opportunities since the close interaction with the teachers enables immediate and detailed feedback. This resonates with what Babb et al. report from industry where the need to meet deadlines had a negative impact on the Scrum teams possibility for reflection-in action during the sprint retrospectives [3].

Our suggestion is to introduce Scrum using workshops and a platform with low technical demands. When students have been introduced to Scrum they can apply the knowledge in a project with a more technical setting, thereby balancing the cognitive load among the process aspects and technical aspects of the project. We have adapted our curriculum accordingly and now offer a Scrum workshop using Lego building blocks as part of the different courses to allow students of all semesters to experience Scrum this way. In the future, every student in the first term will participate in such an exercise and therefore be prepared for the use of Scrum in the second term Software Engineering project.

References

Agility in Dynamic Environments: A Case Study for Agile Development

Simon Brooke and Dina Allswang

Abstract. Continual technological advances require software solutions, products and platforms to be adaptable, extendible and rapidly created to meet market needs.

Software technologies must be interchangeable, environments transparent, solutions maintainable, continuously integrated and delivered with value to the customer.

This paper presents a video software solution in which architecture, design, development, quality and integration were all done in an Agile manner to meet quality requirements, functional requirements, timelines and customer needs, while ensuring adaptability to changes at continuously high quality.

1 Background

In the past, our software development was based on Waterfall methodology. Requirements and design were many months ahead of implementation and coding periods were several months. High-level management decided that Agile could improve development processes. An Agile expert was consulted. Several problematic areas in the development lifecycle were identified:

- Major code rewrites due to late requirement changes
- Several rounds of fixes due to long, serial development/QC
- Integration bugs found too late in the development process
- Functionality not always as intended due to lack of ongoing feedback

2 New Project Using Scrum

An inter-departmental team was selected, consisting of personnel with varying levels of experience and multiple areas of expertise. The Scrum methodology was adopted for the project and various roles were assigned.

The project involved development of 5 components and integrating them into a larger subsystem. High-level architecture was managed using Agile and detailed architecture for each sprint was reviewed with the product owner.
3  **Sprint Activities**

The product owner managed the prioritized backlog in consultation with all stakeholders. For each sprint, the team members declared their capacity and selected tasks from the product owner’s prioritized sprint plan.

Daily stand-up meetings provided close collaboration between team members, to request help and prevent impediments. The team updated progress on the scrum board enabling re-assignment of resources if necessary.

Retrospectives were held to discuss successful methods and improvements.

4  **Continuous Integration**

The components were continuously tested and integrated into the subsystem. Automated tests were run nightly and issues handled daily. This enabled completion of development and QC within sprint scope.

5  **Collaboration with Project Customer**

Detailed design, requirements and implementation were reviewed with the project representative including a demo per sprint enabling timely feedback.

6  **Conclusion**

A measure of the project success was the satisfaction level of the customers. They appreciated the potentially shippable products available at the end of each sprint including a set of expected and high quality, high-priority features enabled by continuous customer involvement and feedback.

**References**

Introducing SafeScrum

Geir Kjetil Hanssen¹, Ingar Kulbrandstad², and Børge Haugset¹

¹ SINTEF ICT, Strindveien 4, 7465 Trondheim, Norway
² Autronica Fire & Security, Haakon VII’s gate 4, 7041 Trondheim, Norway
ghanssen@sintef.no, Ingar.Kulbrandstad@autronicafire.no, borge.haugset@sintef.no

Abstract. Safety-critical systems such as process control- and signal systems play a fundamental role in many important industries and aspects of society. Such systems, including software, must fulfill extreme demands for correct operation and integrity, meaning that development must follow strict standards (in our case IEC 61508), which has to be documented and evaluated by external certification bodies to receive necessary approval. We present recent industrial experience in applying the SafeScrum approach in a complex software organization.

Keywords: Safety-critical systems · SafeScrum · IEC 61508

1 Agile Development of Safety Critical Systems

An increasing part of the functionality of safety-critical systems, e.g. ship-control and fire-detection are implemented as software due to more standardization of hardware. This gives larger and more complex software development projects, which must meet the same strict requirements for safety operation performance as electronically based systems. This calls for efficient and flexible development processes. Present methods are tailored for hardware development, emphasizing low requirements- and design-change frequency, being plan-based and document driven. This gives low requirements change tolerance and high costs (25-50% of total project costs) for producing documentation to prove conformance to standards such as IEC 61508 (process control systems) [1].

2 SafeScrum

To challenge this trend we have proposed the SafeScrum approach [2], which is an extension of the Scrum software development methodology in combination with common XP techniques. Scrum has now been adapted to fit development of safety critical systems and has been harmonized with software lifecycle requirements defined in the IEC61508 standard. Fitting Scrum for this type of development and certification means that it has to be extended with additional activities. For example, code reviews need to be documented with full traceability in such a way that an independent assessor can
verify that all code has been reviewed according to the standard. Another important extension is routines for change impact analysis, meaning that changes in requirements must be followed by an analysis on whether the change may impact the safety function of the system. This must also be fully traceable and documented for external verification. The overall challenge is to utilize the benefits of an agile approach, having it aligned with the standards extensive requirements for documentation while at the same time avoiding too much extra weight.

3 Introducing SafeScrum – Early Lessons Learned

Autronica Fire & Security AS has for over a year introduced SafeScrum in a SIL3 industry project (very high safety performance). Key lessons learned are:

1. **Adaptation and adoption of a radically different process needs change agents:** Although supported by external experts, the shaping of SafeScrum and the change process itself happened as a grass-root movement. Having the ones that are to be using the process implement it themselves is important to establish sufficient motivation and to build necessary hands-on knowledge on a very detailed level.

2. **A radical change costs extra resources and needs support from management:** In our case management strongly supported the change, giving the team trust, freedom and time to try and fail. Without this support and resources it would have been extremely hard to succeed bottom-up.

3. **External support and validation strengthens the change:** Although the change happened bottom-up, external input on methodology and safety assessment (researchers) as well as discussions with the certification body (TÜV in this case) has enabled the team to prioritize change actions, discuss ideas and evaluate the suitability of SafeScrum and compatibility with the IEC 61508 standard.

4. **Tools are as important as processes:** There is a need to provide extensive documentation of compliance with the process. Traditionally this results in a lot of effort spent to produce explicit documentation in addition to the development itself. SafeScrum works very much because tools automate or supports the team in creating this type of information.

5. **Change needs to be done step-by-step:** The new process was started in the simplest possible way with only the core Scrum elements. Eventually, more details were added based on constant evaluation and refinement of the process. This approach also gave the team confidence as they always had a fully working process, which was in-line with the IEC 61508 standard.

References


Revisit – A Systematic Approach to Continuously Improve Agile Practices in Large-scale and Fast-expanding R&D Center

Peng Liu and Yuedong Zhao

Nokia Networks Chengdu Technology Center,
Building A4, Tianfu Software Park, High-tech Zone,
610041 Chengdu, Sichuan, P.R.China
{brian.1.liu,yuedong.zhao}@nokia.com

Abstract. We give an overview of our recent initiative in designing a systematic framework to help teams continuously improve Agile practices in large-scale and fast-expanding R&D centre. The framework is expected to be a closed-loop, light-weighted and sustainable one with positive business impact. To address these requirements, after several rounds of pilot, a “Revisit” framework is figured out, which consists of five major phases: 1) Goal/Problem-driven planning; 2) Light-weight audit; 3) Audit result analysis and reporting; 4) On-demand Competence Development support leveraging resources across the organization; 5) Progress tracking and effectiveness check. Also, a joint-force including a Revisit team, Revisit agents, and ScrumMaster community is setup to ensure the effective execution. Initial results indicate promising evidence this approach can help teams’ quality and efficiency improvement.

Keywords: Agile method · Scrum · Continuous improvement · Competence development · Light-weight audit · Software quality support processes

1 Introduction

Agile, as an effective strategy to make software enterprises more flexible and responsive to changes, has one vital focus on continuous improvement [1]. However, in large-scale software R&D center with teams in different maturity levels, immature teams are most likely to be incompetent and inefficient in inspecting the real barriers in their agile practice and corresponding root causes and therefore adapting by taking effective improvement actions [2]. On the other hand, for matured teams, they may become stagnant, and it is a challenge to reinvigorate these teams in their agile practice to make it more efficient in creating customer values. Moreover, for a fast-expanding R&D centre (e.g., in China, India and Poland), a new challenge is to ensure new hires do not introduce too much “turbulence” to the agile way of working of existing teams. Yet another aspect to consider is employee Competence Development.
(CoDe). The fact is, however, employee CoDe team is most likely a central function not working closely with teams, which may lead to a mismatch between teams’ real training needs and the resources (e.g., training, coaching) provided by the CoDe team.

Facing all above challenges, in Nokia Networks Chengdu Technology Centre (TC), there is a high demand for a mechanism to effectively help teams across the organization to continuously improve their agile practice with major requirements as:

**Requirement 1: Light-weight.** Teams expect low overhead and interruption.

**Requirement 2: Close-loop.** The mechanism shouldn’t stop with identifying problems but should form a closed Plan-Do-Check-Act (PDCA) loop to help team grow.

**Requirement 3: Effectiveness.** CoDe support should be based on problems identified and positive business impact are expected from team.

**Requirement 4: Sustainability.** Good agile practice should sustain, and team should be capable for inspecting and adapting themselves.

## Solution

As a result, in Nokia Network Chengdu TC, after several pilots, we finally figure out the “Revisit” framework as a systematical approach to help teams continuously improve agile practice in large-scale and fast-expanding software powerhouse.

![Revisit framework: An overview](image)

Enabling teams: 1) Revisit team; 2) Revisit Agents; 3) ScrumMaster community

Highlights of lessons learnt are: First, this bottom-up initiative really gains buy-in from teams; second, refresh on Agile principle and Scrum practice is a must in fast-expanding R&D to avoid pitfalls like mini-waterfall; third, on-demand CoDe based on issues identified in Revisit is more effective in achieving positive business impact.

## References

Applying Randori-Style Kata and Agile Practices to an Undergraduate-Level Programming Class

Chitsutha Soomlek

Department of Computer Science, Faculty of Science, Khon Kaen University, Khon Kaen, Thailand
chisutha@kku.ac.th

Abstract. This study adapted and applied a traditional randori-style kata to a web programming class, in order to help undergraduate students improve their programming skills. The process in the traditional randori-style kata was modified to match the nature of XML. The results indicated that the modified randori-style kata is an effective method for the programming course and the students to enhance their programming skills. The activity also helped the students to repetitively review and reapply theories and knowledge to programming problems. The kata can stimulate the learning system, enhance self-confidence, and improve the relationships among the students.

Keywords: Randori-style kata · Coding dojo · Kata · Learning systems · Programming education

1 Introduction

A kata, also called coding dojo, is an activity designed for programmers to improve their coding skills through practices, repetition, and knowledge sharing [1]. A kata includes agile practices; e.g., pair programming, Test-Driven Development (TDD), Continuous Integration (CI), refactoring, and timeboxing; in the activity [1]. Swamidurai et al. indicated that using peer code review in the TDD context is even more effective for a programming practice than the traditional pair programming [2].

2 Procedure and Methods

In order to help our undergraduate students to improve their coding skills, a modified version of randori-style kata was created and integrated in a web programming class at the Department of Computer Science, Khon Kaen University, Thailand. The class was designed for 3rd-year and 4th-year students and focused on XML, XML-based technologies, Web technologies (e.g. HTML, XHTML, and JavaScript), and their applications (e.g. AJAX and Web services). The modified kata was assigned as one of the class activities in the Fall 2014 semester. There were 79 3rd-year students registered in this class. There were one instructor and two teaching assistants.
In the traditional randori-style kata, TDD is primarily used. However, writing a test before writing a code in XML is different from other programming languages that already have corresponding testing frameworks, e.g., Java and JUnit, PHP and PHPUnit, etc. Therefore, instead of writing a failing unit test on a testing framework, Document Type Definition (DTD), XML schema, and test cases were employed. For example, the instructor introduced a programming problem, a corresponding DTD/XML schema, and explained the acceptance criteria with examples to the students. The students had to write a well-formed and valid XML document according to the definitions provided in the DTD/XML schema. Sometimes, an XML document was given; then, the students had to write a DTD/XML schema that is complied with the XML document while keeping the document well-formed and valid. After the students were familiar with how to write a DTD/XML schema, they had to write a definition in DTD/XML schema first, and then, they had to create corresponding XML tags. Moreover, the instructor occasionally gave subtle hints, strategies, direct instructions, or changes of requirements during the practice sessions.

In the traditional kata, audiences are not allowed to disturbed the pilot and co-pilot, i.e. paired programmers. In this class, students were allowed to deliberate, share, and discuss their ideas with the class.

3 Results

This class was evaluated through three sets of questionnaires with open-end questions and interviews. 79 students registered in this course.

26 students participated in the modified kata \( \geq 2 \) times. 22 students participated only once. 25 students did not participate in the kata. 6 people did not answer the questionnaires. 100\% of participants agreed that the activity was an excellent activity for reviewing what they had learnt from the class, can help them understand XML, and can improve their coding skills. The activity can also motivate the students to hone their coding skills. Moreover, the modified randori-style kata can improve self-confidence and relationship among the students. However, the non-participants stated several reasons why they did not participate in the modified kata, for example, the session ended before their turns and they did not know how to solve the problems.

The modified randori-style kata could be used as an alternative method for programming practices. It can be applied to any programming courses and used for training purposes. Moreover, there are many possibilities to enhance the activity such as having multiple groups of kata working on the same programming problem.

References


Continuous Strategy Process in the Context of Agile and Lean Software Development

Tanja Suomalainen¹ and Jenni Myllykoski²

¹ VTT Technical Research Centre of Finland
Kaitoväylä 1, P.O. Box 1100, FI-90571 Oulu Finland
{Tanja.suomalainen}@vtt.fi
² Oulu Business School Pentti Kaiteran katu 1,
P.O. BOX 4600, FI-90014 University of Oulu Finland
{Jenni.myllykoski}@oulu.fi

Abstract. This extended abstract introduces an ongoing research which elaborates the concept of continuous strategy with an aim to better address the contemporary challenges of strategy process within the field of software development. Theoretically the research draws on the strategy process and practice -literature and on the literature of agile and lean software development when conceptualizing continuous strategy. Empirically the research examines the emerging challenges in software companies’ strategy process when they are pursuing continuous practices in software development. The tentative analysis revealed that the practices that stem from the more traditional, structured strategy process are not compatible with the more “continuous planning”-oriented organisational practices. Thus, there seems to be pressure for developing more continuous strategy process and practices. Furthermore, in other organisational processes, especially in team level, the practices relating to continuity are more successfully applied, than in strategic level.

Keywords: Strategy process · Continuous planning · Software development

1 Introduction

In recent few years, the adoption of agile and lean development practices has increased remarkably in Information and Communication Technology (ICT) industry. Even though many companies have succeeded in adopting these practices, even more innovative approaches that support continuous practices throughout the organisation are needed [1]. For example, [2] emphasize continuous integration between software development and its operational deployment as well as continuously assessing and improving the link between business strategy and software development. Similarly, there is a clear need for continuous planning in which plans are dynamic open-ended artifacts that evolve in response to changes in the business environment. However, the continuous software development practices appear to be ill assorted with the traditional, rationalistic view of strategy process. Whereas the continuous software development stresses the importance of real time actions and continuous change [1], the
rationalistic view of strategy process is very static and future-oriented. It relies on the assumption of relatively predictable business environment that allows the rational managers [3] to create a long term future plan based on systematic scanning and positioning [4], and subsequently to implement it while having sufficient control over the consequences of actions [3]. The usefulness of such theories in the context of software development is questionable.

2 Research Method

We argue that continuous software development and planning calls for flexibility also in the strategy process and therefore the purpose of this research is to introduce and empirically investigate the concept of continuous strategy. The concept is rooted on the emerging literature on strategy process and practice that acknowledges continuous change as the inherent feature of reality and views strategy as fluxing, improvisational and temporal process [3, 4]. In addition, we draw on the literature on continuous software development, with an aim to connect the concept of continuous strategy with the practice of agile and lean software development. The research builds on a qualitative multi case study [5] in which data is collected through several interviews (altogether 22 at the moment) and analysed with the help of NVivo tool. The research is done in Digile’s Need for Speed (N4S) research program (http://www.n4s.fi/en/).

The tentative analysis of an interview data revealed that the practices stemming from the “traditional”, structured strategy process are not compatible with the more continuous planning practices within the organisation. Thus, there seems to be pressures for developing continuous strategy process and practices. Furthermore, in other organisational processes, especially in team level, the practices relating to continuity are more successfully applied, but in strategic level continuity is not yet achieved.

References

Automatizing Android Unit and User Interface Testing

Juha-Matti Vanhatupa and Mikko Heikkinen
PacketVideo Finland Oy, Hallituskatu 8, FI-33200 Tampere, Finland
{vanhatupa,mheikkinen}@pv.com

Abstract. In this paper we describe our efforts moving towards automatized testing in Android application development. With sophisticated combination of Android development tools and Perl scripts, we have automatized parts of the testing produce. Using the built system, unit tests are built and run automatically each time when a new build is created. In addition, we are automatizing user interface testing using Android uiautomator tool. In this automatization process we encountered several technical issues, for which Android development tools do not offer ready-made solutions, and we had to implement solutions for those.

Keywords: Test automation · Android · graphical user interface testing

1 Introduction

Mobile applications are designed to run on smartphones, tablet computers and other mobile devices. They are a challenging target for testing, many of them are web applications using resources from the Web, and they can also use data from sensors of the device. In addition they feature many different versions of operating systems and software frameworks customized for different devices [4].

The use of unit testing is generally low in mobile development community [1]. Approximately 20% of developers stated that their companies do not use any unit testing, and when performed it was mostly done manually. In addition, graphical user interface testing is challenging to automatize, and it is mostly done manually [1]. It was automatized only in 3% of the cases.

We have automatized executing unit tests with combination of Perl scripts and Android build tools. In addition we convert manual user interface (UI) tests into automatized tests using uiautomator framework [3]. The target application of this work was Twonky Beam [2], an app that lets users discover and transfer online video content from a tablet or mobile device to a television.

2 Test Automatization Approach

The purpose of automatizing unit testing is to ensure these tests are executed with each build. Natural solution was to include compiling and running unit tests
into build scripts of Twonky Beam. So they are executed, when a new feature or bug fix is pushed to the remote stream in version control and a build is launched on Jenkins. Scripts used for compiling Twonky Beam are written in Perl and we used it also for unit test automatization.

Perl has many built-in functions for shell programming and it can call operating system facilities. This enabled compiling projects (application project and unit test project), and starting Android emulator, and running unit tests on it.

We started automatization of UI testing using uiautomatorviewer tool. The tool enables taking screenshots of UI and viewing layout hierarchy of UI components. Then we implemented UI test cases. Manual UI tests are converted into uiautomator tests. Almost all of our 80 UI tests are possible to automatize using uiautomator, excluding only test cases, which required rebooting the device, or manual switching on of wireless networks.

When implementing uiautomator test cases, we encountered a problem with WebView component. Uiautomator cannot gather info about UI components inside it, since it sees WebView only as one component. We resolved the problem using coordination calculation algorithm, which calculates correct coordinates based on pixel density of the device and made a mouse click using those.

Another issue in UI testing was caused by different localization versions of Twonky Beam application (English and Japanese), because uiautomator uses text strings of UI components to locate them. To avoid failures in string comparison we included an extra step to the build process of uiautomator project. In this step files containing localized strings are copied into uiautomator project. When text string of UI component is required in the project, a separate function is called and the string is fetched from the correct file of localized strings.

3 Conclusions

Android framework offers many valuable tools, e.g. uiautomator is an excellent tool for UI test automatization. Although, as we described special cases in UI testing have to be handled separately. Overall, test automatization can save time and resources. The implemented approach can be used with other Android applications also.

References

Author Index

Abrahamsson, Pekka 52
Ahmad, Muhammad Ovais 178
Alégroth, Emil 361
Allswang, Dina 363
Ammälä, Mikko 205
Arcelli Fontana, Francesca 3
Aro, Ismo 329
Bajwa, Sohaib Shahid 52
Barroca, Leonor 64
Berger, Christian 15
Bjarnason, Elizabeth 27
Bjerke-Gulstuen, Kristian 239
Boekhorst, Laurens 248
Borg, Markus 27
Bosch, Jan 218, 350
Brooke, Simon 363
Burden, Håkan 361
Chmelár, Martin 302
Claassen, Arjan 248
Counsell, Steve 129
Destefanis, Giuseppe 129
Diebold, Philipp 40, 345
Dingsøyr, Torger 239
 Dönmez, Denniz 193
Doss, Osama 347
Eckstein, Jutta 339
Eklund, Ulrik 15
Engström, Emelie 27
Ericsson, Morgan 361
Fabijan, Aleksander 350
Fontana, Rafaela Mantovani 199, 310
Fraser, Steven 334, 329
Fridstrom, Josh 294
Giardino, Carmine 52
Gregory, Peggy 64
Grote, Gudela 193
Hammouda, Imed 361
Hanssen, Geir Kjetil 365
Harzl, Annemarie 352
Haugset, Børge 365
Heikilä, Ville T. 93
Heikkinen, Mikko 373
Hokkanen, Laura 81
Holyer, Steve 339
Hunt, Johanna 254
Ilieva, Sylvia 230
Itkonen, Juha 93
Jacques, Adam 294
Järvinen, Janne 334
Karvonen, Teemu 354
Kassab, Mohamad 129
Kelly, Tim 347
Kettunen, Petri 205
Kilpela, Kurt 294
Kirkvainen, Antti 339
Kivioja, Henri 329
Knauss, Eric 361
Krchnák, Jan 302
Kulbrandstad, Ingar 365
Kuvaja, Pasi 212, 354
Larsen, Diana 339
Larsen, Emil Wilk 239
Lassenius, Casper 356
Laukkanen, Eero 356
Lehtinen, Timo O.A. 93
Lenarduzzi, Valentina 105
Lindgren, Eveliina 117
Liu, Peng 367
Lundh, Erik 329, 334
Lunesu, Ilaria 105
Lwakatare, Lucy Ellen 212
Malucelli, Andrea 199, 310
Marchesi, Michele 129
<table>
<thead>
<tr>
<th>Author</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marshall, Stuart</td>
<td>141</td>
</tr>
<tr>
<td>Martini, Antonio</td>
<td>218</td>
</tr>
<tr>
<td>Matinlassi, Mari</td>
<td>262</td>
</tr>
<tr>
<td>Matta, Marta</td>
<td>105</td>
</tr>
<tr>
<td>Md. Rejab, Mawarny</td>
<td>141</td>
</tr>
<tr>
<td>Meyer Jr., Victor</td>
<td>199</td>
</tr>
<tr>
<td>Münch, Jürgen</td>
<td>117, 154</td>
</tr>
<tr>
<td>Myllykoski, Jenni</td>
<td>371</td>
</tr>
<tr>
<td>Noble, James</td>
<td>141</td>
</tr>
<tr>
<td>Nybom, Kristian</td>
<td>166</td>
</tr>
<tr>
<td>Oivo, Markku</td>
<td>178, 212, 354</td>
</tr>
<tr>
<td>Olsson, Helena Holmström</td>
<td>350</td>
</tr>
<tr>
<td>Ortu, Marco</td>
<td>129</td>
</tr>
<tr>
<td>Ostberg, Jan-Peter</td>
<td>40</td>
</tr>
<tr>
<td>Pareto, Lars</td>
<td>218</td>
</tr>
<tr>
<td>Partanen, Jari</td>
<td>205, 262</td>
</tr>
<tr>
<td>Porres, Ivan</td>
<td>166</td>
</tr>
<tr>
<td>Poupko, Avraham</td>
<td>270</td>
</tr>
<tr>
<td>Power, Ken</td>
<td>329, 334</td>
</tr>
<tr>
<td>Reinehr, Sheila</td>
<td>199, 310</td>
</tr>
<tr>
<td>Rising, Linda</td>
<td>329, 334</td>
</tr>
<tr>
<td>Rissanen, Olli</td>
<td>154</td>
</tr>
<tr>
<td>Rodríguez, Pilar</td>
<td>178</td>
</tr>
<tr>
<td>Rosenberg, Shlomi</td>
<td>279</td>
</tr>
<tr>
<td>Salah, Dina</td>
<td>64</td>
</tr>
<tr>
<td>Sarkela, John</td>
<td>294</td>
</tr>
<tr>
<td>Schneider, Matti</td>
<td>287</td>
</tr>
<tr>
<td>Sennersten, Charlotte</td>
<td>224</td>
</tr>
<tr>
<td>Sharp, Helen</td>
<td>64</td>
</tr>
<tr>
<td>Siverland, Susanne</td>
<td>224</td>
</tr>
<tr>
<td>Slany, Wolfgang</td>
<td>352</td>
</tr>
<tr>
<td>Smeds, Jens</td>
<td>166</td>
</tr>
<tr>
<td>Šmiřák, Roman</td>
<td>302</td>
</tr>
<tr>
<td>Soomlek, Chitsutha</td>
<td>369</td>
</tr>
<tr>
<td>Sorje, Olli</td>
<td>339</td>
</tr>
<tr>
<td>Stålhane, Tor</td>
<td>239</td>
</tr>
<tr>
<td>Stavru, Stavros</td>
<td>230</td>
</tr>
<tr>
<td>Steghöfer, Jan-Philipp</td>
<td>361</td>
</tr>
<tr>
<td>Suomalainen, Tanja</td>
<td>371</td>
</tr>
<tr>
<td>Taibi, Davide</td>
<td>105</td>
</tr>
<tr>
<td>Taylor, Katie</td>
<td>64</td>
</tr>
<tr>
<td>Tonelli, Roberto</td>
<td>129</td>
</tr>
<tr>
<td>Tramontini, Ramon</td>
<td>310</td>
</tr>
<tr>
<td>Tripathi, Nirmaya</td>
<td>178</td>
</tr>
<tr>
<td>Tureček, Tomáš</td>
<td>302</td>
</tr>
<tr>
<td>Unterkalmsteiner, Michael</td>
<td>27</td>
</tr>
<tr>
<td>Väänänen-Vainio-Mattila, Kaisa</td>
<td>81</td>
</tr>
<tr>
<td>Vanhatupa, Juha-Matti</td>
<td>373</td>
</tr>
<tr>
<td>Virtanen, Risto</td>
<td>93</td>
</tr>
<tr>
<td>Wagner, Stefan</td>
<td>40</td>
</tr>
<tr>
<td>Walter, Marcelo</td>
<td>310</td>
</tr>
<tr>
<td>Wang, Xiaofeng</td>
<td>52</td>
</tr>
<tr>
<td>Wernersson, Roger C.S.</td>
<td>224</td>
</tr>
<tr>
<td>Wild, Werner</td>
<td>329, 334</td>
</tr>
<tr>
<td>Wilson, Alexander</td>
<td>319</td>
</tr>
<tr>
<td>Wirfs-Brock, Rebecca</td>
<td>329, 334</td>
</tr>
<tr>
<td>Zanoni, Francesco</td>
<td>3</td>
</tr>
<tr>
<td>Zanoni, Marco</td>
<td>3</td>
</tr>
<tr>
<td>Zendler, Ulrich</td>
<td>40</td>
</tr>
<tr>
<td>Zhao, Yuedong</td>
<td>367</td>
</tr>
</tbody>
</table>