Report from Dagstuhl Seminar 12442

Requirements Management – Novel Perspectives and Challenges

Edited by

Jane Cleland-Huang[1], Matthias Jarke[2], Lin Liu[3], and Kalle Lyytinen[4]

1 **DePaul University – Chicago, US,** `jhuang@cs.depaul.edu`
2 **RWTH Aachen and Fraunhofer FIT, DE,** `jarke@cs.rwth-aachen.de`
3 **Tsinghua University Beijing, CN,** `linliu@tsinghua.edu.cn`
4 **Case Western Reserve University – Cleveland, US,** `kalle@case.edu`

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 12442 "Requirements Management – Novel Perspectives and Challenges". Changes in computational paradigms and capabilities that draw upon platform strategies, web services, and virtualization of both application services and development platforms have significant implications for views of modularity and requirements evolution, complexity of RE tasks, and the economics of system development and operations. The aim of the seminar was to bring together experts from multiple fields to discuss models and theories around these changes. Three key challenges and associated solution ideas were addressed, namely (1) to better deal with context changes and business goal management to reduce the "black swan" rate of badly failed large projects, (2) to exploit recent theories of technological and institutional evolution to understand better how to control complexity and leverage it for innovation at the same time, and (3) the demand for runtime re-organization of existing large-scale systems with respect to new operational goals such as energy efficiency. Future RE must see itself as the marketplace where responsibility for all these complexities and evolutionary steps is traded.

Seminar 28.–31. October, 2012 – www.dagstuhl.de/12442
1998 ACM Subject Classification D.2.1 Requirements, D.2.2 Design Tools and Techniques, D.2.11 Software Architectures
Keywords and phrases requirements engineering, system complexity, software evolution, socio-technical systems
Digital Object Identifier 10.4230/DagRep.2.10.117
Edited in cooperation with Anna Hannemann (RWTH Aachen, DE)

1 Executive Summary

Matthias Jarke

Since its inception in the 1970s, much of the research in requirements engineering (RE) has focused on the development of formal notations and protocols to represent requirements and to analyze their properties, such as consistency, correctness, completeness, and validity. Some work has analyzed the impacts of these requirements on downstream development tasks (e.g., traceability), or managing and reconciling conflicts in the requirements process.

Much of requirements research has also assumed that the scope of RE is isolated to a specific project or even a specific stage of that project. The demand for a shift in focus is dictated by changes in computational paradigms and capabilities that draw upon platform strategies, web services, and virtualization of both application services and development platforms. These trends have significant implications for views of modularity and requirements evolution, complexity of RE tasks, and the economics and costs related to application and service use and development. The aim of the seminar was to bring together experts from multiple fields to discuss models and theories around these changes, focusing on a series of interrelated question such as:

- How to theorize and study complexity within RE tasks?
- What theoretical perspectives can inform how and why requirements knowledge evolves as it is generated, validated, and distributed?
- How do requirements, system evolution, and environmental change interact?
- How do different types of knowledge interact to shape requirements and their evolution?
- What are the origins and flows of influence of requirements knowledge? How can non-linear influences be effectively managed in RE evolution?
- What is the effect of speed and scale in requirements processes?
- What is the role of goals and constraints and their complex interactions in RE?

In particular we sought better integration of theories of socio-technical system evolution, distributed cognition, models of RE and design knowledge and their economic effects, the impact of strategy and related knowledge endowments in RE processes (e.g., explorative vs. exploitative processes of requirements discovery), and the role of ambiguity, uncertainty and complexity in managing requirements knowledge. Attention was also placed on new research approaches and methods that can be brought to bear in addressing these problems. The seminar thus built and expanded on some of the critical themes that had been brought up five years earlier in two NSF-sponsored workshops in Cleveland [5] and Dagstuhl [4], [3]. The seminar brought together 33 researchers (exactly one third female) from 12 countries in four continents, with 22% industry participation. Participants felt that this unusually high diversity together with a good mix of junior and senior people of different disciplines, interests and expertise contributed strongly to lively and fruitful discussions. Several cooperative projects have emerged from these discussions. Selected results of the discussions and presentations will be published in a special issue of the ACM Transactions on Management Information Systems in 2014.The program of the seminar was organized into four panels with plenary talks and discussion, five parallel working groups with central reporting, and a final reflection session. With the parallel Dagstuhl seminar on "Foundations and Challenges of Change and Evolution of Ontology" we moreover organized a crossover plenary panel session in which we tried to converge to a better mutual understanding of the different perspectives on Evolution in AI and RE and explored possibilities for future cooperation. Several individual researchers later got together to agree on specific cooperative research. In the final reflection session, the main results, issues and challenges, also taking into account the ontology perspective, can be summarized as follows.

Jackson and Zave [2] have formulated an AI-inspired formalization of the traditional RE viewpoint as a kind of model-based diagnosis: Given a set of domain assumptions D and a set of requirements R, find a suitable specification S such that $S, D \Rightarrow R$. In RE research, the R have often been interpreted as goals, to be refined and satisfied in some extended $AND - OR$ graph structure.

From a social science and business informatics perspective, however, requirements engineering (RE) is in essence a boundary spanning task between the developers and the other stakeholders (users, management, regulators, ...) concerning the goals, functions and constraints of a system. The traditional viewpoint, where RE is just seen as an "early phase" (resulting in a contract) and the "last phase" (where acceptance testing takes place) is far too narrow. The following citation by Robert Glass wonderfully characterizes the situation RE has entered since the turn of the century: *"Walking on water, and programming according to specifications is easy – as long as both of them are frozen"* At least three key challenges to research and practice were identified in the seminar, together with counter-strategies where promising first steps for solutions were observed: Firstly, *large-scale projects* encounter changes in D, R, and the technology underlying S is shifting. As a consequence,

- 90% of these projects run over budget and time (this is similar to other big engineering projects, so not a drama in itself)
- One sixth so-called *Black Swan* projects show budget overruns of 70% and time overruns of 200% (this is true only for 1% of other engineering projects, so this is a real drama of software engineering).

A central cause is politically motivated over-ambitious goals with systematic under-estimation of the nature and scope of requirements, budget, and cost both on the side of customers and vendors, as well as poor change tracking. As a consequence, we strongly recommend to not just consider goals of stakeholders but also social structures and strategic dependencies in initial system analysis. Moreover, customer and other stakeholder requirements must be continuously monitored during the development process (and sometimes beyond). To ensure product and process compliance and effectively assess the impact of change, requirements traceability should be focused by using trace patterns to maintain transparency and keep the monitoring effort acceptable and feasible.

Secondly, we need architectural mechanisms that *constrain, but also leverage complexity.* In the seminar, John King pointed out the difference between "complicated" and "complex" problems. Complicated problems can be solved by experienced, highly competent engineers with foreseeable effort. In contrast, complex problems can only be explored with uncertain results; thus, taking on a project that tries to solve a complex problem in one shot is bound to lead to disaster – and apparently, it is exactly the tendency to take on such nice-sounding complex projects that leads to the unusually high share of Black Swans in software projects. Theories like Arthur's theory of Technology Evolution [1] or Thornton's theory of institutional evolution [6] were cited in the seminar as showing a way forward, which we can also observe in practice. Platform strategies offer complicated but manageable base solutions that are now being offered both by open source communities and by big players in different sectors, such as IBM, Google, Facebook, and mobile phone vendors/operators. With a uniform infrastructure, they limit complexity. But by enabling innovation at the margin, e.g. end-user developed app's, they at the same time also leverage new complexity at the higher level. The easy entry, combined with ruthless selection of a very small percentage of truly successful apps, then offers a hotbed of complex evolutionary change. Which eventually will grow into, or be replaced again by yet another layer of platforms, as pointed out by both Brian Arthur and the earlier book by Thomas Friedman "The World is Flat". Beyond such market selection mechanisms, software vendors employ various mechanisms to participate in this game in a more controlled way. We mention here the now broad area of software product families, but also Google's 70 : 20 : 10 work rule where employees are free to spend a significant part of their work time on their own ideas, thus fostering continuous internal innovation. Methods for runtime requirements monitoring and requirements mining from usage patterns can be important contributions of the RE field in this context. Last not least, the future will not reduce the challenges of complexity and evolution. Our seminar understood information systems as socio-technical systems, but in fact many of today's systems are neither truly social nor truly technical. From a social perspective, there is the new question for sustainability of systems, with the demand for re-optimization from the viewpoint of user rights (e.g. asymmetric information and market powers, privacy, data ownership, copyright vs. freedom of information), energy efficiency, and environmental footprint. From a technical perspective, the explosive expected growth of Cyberphysical Systems (Internet of Things) in business, engineering and science is not just an approach to monitor and actuate at a much more fine-grained level, but a significant source of more complexity and evolutionary challenges. Generating and implementing e.g. the visions of smart cities is just but one example. Rather than just talking grand new visions here, methods for how to get there step-by-step in an "only" complicated way – without exposing whole city to the chaos caused by over-ambitious "complex" systems – are urgently needed.In our world of more and more ubiquitous computing, where the impact and complexity of systems continuously seem to grow, *RE is the marketplace where responsibility is traded*, as communication, mutual understanding, and transparent well-structured information management are at the heart of this field.

References

1 B. Arthur (2009). The Nature of Technology: What it is and how it Evolves. Free Press.
2 M. Jackson, P. Zave (1995): Deriving specifications from requirements: an example. Proc. 17th ICSE.
3 M. Jarke, K. Lyytinen, eds. (2010): High Impact Requirements Engineering. Special Issue, Wirtschaftsinformatik/BISE 52, 3.
4 M. Jarke, P. Loucopoulos, K. Lyytinen, J. Mylopoulos, W.N. Robinson (2011): The brave new world of design requirements. Information Systems 36(7): 992–1008.
5 K. Lyytinen, P. Loucopoulos, J. Mylopoulos, W.N. Robinson, eds. (2009). Design Requirements Engineering – A Ten-Year Perspective. Springer LNBIP 14.
6 P. Thornton, W. Occasio, M. Lounsbury (2012). The Institutional Logics Perspective: A New Approach to Culture, Structure, and Process. Oxford University Press.

2 Table of Contents

3 Overview of Talks

3.1 Requirements for Digital Infrastructure Innovation: Three Broad Strategies for Organizations

Nicholas Berente (University of Georgia, US)

Trends in requirements engineering are shifting from a focus on discrete software projects for a particular user community in a particular organization to broad co-evolving and interdependent ecosystems of hardware, software, communications, social practices, business processes, and organizational structures ([7]). This shift is reflected in the increased emphasis on architectures, platforms, and integration that enable ongoing emergent and unpredictable phenomena ([5]; [7]). Contemporary artifacts are increasingly infrastructural, and innovation on digital infrastructures is a fundamentally different sort of thing from the discrete software engineering context around which the requirements engineering discipline grew up. According to Tilson, Lyytinen and Sorensen:

"digital infrastructures can be defined as shared, unbounded, heterogeneous, open, and evolving sociotechnical systems comprising of an installed base of diverse information technology capabilities and their user, operations, and design communities." ([11] p. 748-749, based on [6])

Many organizations – of a variety of different stripes – are looking to participate in and capitalize on digital infrastructure innovation. Organizations do this for a variety of reasons, including:

- Direct Return (examples: Microsoft Office; Apple's platform) – ownership of infrastructures such as platforms as a key element of their product strategy, and associated competitive and profitability issues ([12]);
- Indirect Return (examples: Red Hat Linux; Yahoo! and Hadoop) – building competencies and access to resources which enable indirect business models associated with competitiveness in other domains or complementary products ([4]; [2]);
- Infrastructural Stewardship (examples: U.S. Cyberinfrastructure Centers; Apache and Hadoop) – the organization's mission is to tend to and guide (i.e. "steward") the emergence of a particular digital infrastructure domain ([2]).

Although organizations do look to participate in digital infrastructure innovation, this is not so simple a task. Digital infrastructures are complex and emergent , which poses a number of challenges to their design and management ([6]; [12]). It is simply not possible to specify requirements for an entire infrastructure because of this complexity, and also because of the open, layered generativity that accompanies digital innovation ([13]; [11]). Given this complexity, at this point there is no clear guidance for how organizations who wish to participate in digital infrastructure innovation should go about their requirements activity to define the scope of such innovation.

To begin addressing this situation, we draw upon research into U.S. cyberinfrastructure centers [2] and how they handle requirements for infrastructural innovation. We find three broad strategies that go from less to more centrally controlled: (1) a problem solving strategy; (2) a portfolio & market strategy; and (3) an institutional shift strategy. Next we briefly address the challenges of digital infrastructure innovation and present these three strategies with brief examples from our research. We conclude with a discussion on the merits of the portfolio and market strategy for organizations wishing to engage in digital infrastructure innovation and motivate the need to understand what requirements for this form of innovation look like.

References

1 Berente, N., Claggett, J., Howison, J. Knobel, C. and Rubleske, J., (2012) "Managing CI Centers: An Agenda for Organizational Scholarship and Cyberinfrastructure Innovation," report on the workshop: 'Managing CI Centers.' Available at SSRN (August 14, 2012):http://ssrn.com/abstract=2128872.

2 Dahlander, L. and Magnusson, M. How do Firms Make Use of Open Source Communities? Long Range Planning 41, 6 (2008), 629–649.

3 Edwards, P., Jackson, S., Bowker, G., and Knobel, C. (2007). Report to the NSF of a Workshop on "History and Theory of Infrastructures: Lessons for new scientific infrastructures." University of Michigan, School of Information.

4 Fitzgerald, B. (2006). The transformation of Open Source Software. MIS Quarterly, 30(4).

5 Hansen, S., Berente N. and Lyytinen, K., (2009) "Requirements in the 21st Century: Current Practice & Emerging Trends," in Lyytinen, Loucopoulos, Mylopoulos, Robinson eds, Design Requirements Engineering: A Ten-Year Perspective, Springer-Verlag (Lecture Notes in Business Information Processing Series Vol.14), 2009.

6 Hanseth, O., and Lyytinen, K. (2010). Design theory for dynamic complexity in information infrastructures: the case of building internet. Journal of Information Technology, 25, 1–19.

7 Jarke, M. (2009) "On Technology Convergence and Platforms: Requirements Challenges from New Technologies and System Architectures," in Lyytinen, Loucopoulos, Mylopoulos, Robinson eds, Design Requirements Engineering: A Ten-Year Perspective, Springer-Verlag (Lecture Notes in Business Information Processing Series Vol.14), 2009.

8 Mitra, S. (2005). Information technology as an enabler of growth in firms: An empirical assessment. JOURNAL OF MANAGEMENT INFORMATION SYSTEMS, 22(2), 279–300.

9 Ribes, David, and Finholt, T. A. (2009). The Long Now of Technology Infrastructure: Articulating Tensions in Development. Journal of the Association for Information Systems, 10(5), 375–398.

10 Star, S., and Ruhleder, K. (1996). Steps toward an ecology of infrastructure: Design and access for large information spaces. INFORMATION SYSTEMS RESEARCH, 7(1), 111-134.

11 Tilson, D., Lyytinen, K., and Sorensen, C. (2010). Digital Infrastructures: The Missing IS Research Agenda. INFORMATION SYSTEMS RESEARCH, 21(4), 748-759. doi:10.1287/isre.1100.0318

12 Tiwana, A., Konsynski, B., and Bush, A.A. 2010. "Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics," Information Systems Research (21:4), pp 685–687.

13 Yoo, Y., Henfridsson, O., and Lyytinen, K. (2010). The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research. Information Systems Research, 4(21), 724–735.

3.2 Models of Institutional Evolution for Requirements Engineering

Nicholas Berente (University of Georgia, US)

Requirements engineering as a discipline is fundamentally concerned with change. Certain sociotechnical practices, however, are deeply entrenched in organizational contexts and, as a result, can be quite difficult to change. As the quotes above indicate, practices can be reinforced from a number of sources: ,"the way we do things around here," implies cognitive economizing; professional pressures imply norms that are driven by identities; and compliance with government mandates implies regulatory pressures. These three forces for deeply entrenching sociotechnical practices are often described as the "pillars" of institutional analysis ([8]). Cognitive, normative, and regulatory sources for institutional persistence and stability.

Sometimes people resist just because they don't want to change. Other times, however, users resist change efforts because there are powerful 'social' forces (of the cognitive, normative, and regulatory variety) that may be reinforcing the existing situation. The former may just be an issue of convincing the person to change and perhaps appeal to their self-interest, whereas the latter may require substantially rethinking the situation. In these situations that are more deeply at odds, it is sometimes useful to investigate whether the "institutional logics" ([5]) that guide the action are consistent between the local context and the change. Institutional logics are the symbolically shaped goals and assumptions implied by patterns of action in a particular domain ([9]). Institutional logics embody the "rules of the game" so to speak ([2]). Organizational contexts are institutionally plural ([6]) and are rife with multiple, often contradictory institutional logics. In cases where the resistance to change is more deeply grounded in contrasting goals, values and assumptions about particular actions that might undermine user identities, the institutional logics of the local context and the change are contradictory ([2]). When business analysts and software engineers go about eliciting requirements for a sociotechnical change, it is important understand those "social" forces that may lead people to resist what appears to be a perfectly reasonable change from the perspective of the development team.

Just as some of these institutional forces tend to reinforce existing situation, new institutional pressure can also help drive change in ways consistent with prevailing or dominant institutions. Organizational actors respond to coercive, normative, and mimetic mechanisms for change ([3]). Thus we can see institutional pressure as simultaneously either a force for stability and a force for change of existing sociotechnical patterns of action. The clash of contradictory institutions is a key driver for bringing about institutional change ([7]). Individuals who draw upon alternative logics to change existing institutions are often described as "institutional entrepreneurs" ([4]; [9]).

Given the stability of organizational contexts borne of entrenched institutions, how can requirements professionals thus navigate this institutional landscape of persistence an isomorphic change? In this essay, we briefly introduce the recent model for institutional evolution following the new "institutional logics perspective" of institutional change ([9]) and reflect on how this model may be relevant for requirements engineering.

References

1 Berente, N. and Yoo, Y. (2012) "Institutional Contradictions and Loose Coupling: Post-Implementation ofNASA's Enterprise Information System," Information Systems Research, Vol.23, No. 2.

2 Bourdieu, P. and Wacquant, L.J.D. (1992) An Invitation to Reflexive Sociology, University of Chicago Press.
3 DiMaggio, P.J., and Powell, W.W. (1983). The Iron Cage Revisited – Institutional Isomorphism andCollective Rationality in Organizational Fields. American Sociological Review, 48(2), 147–160.
4 DiMaggio, P. (1988) "Interest and Agency in Institutional Theory," In L. Zucker (ed) Institutional Patternsand Organizations,Balinger Pub.
5 Friedland, R., and Alford, R.R. (1991) "Bringing Society Back In: Symbols, Practices, and InstitutionalContradictions," in Powell, W.W., and DiMaggio, P.J. eds (1991), The New Institutionalism inOrganizational Analysis, The University of Chicago Press, 1991.
6 Kraatz, M.S. and Block, E.S. (2008) "Organizational Implications of Institutional Pluralism." In R. Greenwood, et al (eds.) Handbook of Organizational Institutionalism, London: Sage
7 Seo, M.G. and Creed, W.E.D. (2002) "Institutional contradictions, praxis, and institutional change: Adialectical perspective." Academy of Management Review, v. 27 issue 2, 2002, p. 222.
8 Scott, W.R. (2008) Institutions and Organizations, third edition, Sage Publications, 2008.
9 Thornton, P.M., Ocasio, W., and Lounsbury, M. (2012) The Institutional Logics Perspective: A New Approach to Culture, Structure, and Process, Oxford University Press.

3.3 Has Time Stood Still in Requirements Engineering?

Richard Berntsson Svensson (Lund University, SE)

Since the inception of requirements engineering in the 1970s, a major shift has taken place. This shift is not only related to, e.g. changes in globalization, but also changes in economic activities and the merge of traditional industries and technology industries. Just as the industrial revolution replaced agriculture as the dominant economic activity, the 'creativity age' is replacing the 'information age' as the next dominant global economic activity. Although the importance of creativity in requirements engineering is argued, both by empirical evidence and that creativity has received more attention in requirements engineering research in the last couple of years, relatively little requirements engineering research has addressed creativity. I believe that economic and market trends imply that requirements engineering will have to become significantly more creative to realize the potential of future software applications. These changes lead to that the complexity and size of software-intensive systems continues to increase. Hence, scaling software in a controlled and efficient way may become a crucial competitive advantage. How many requirements can an industrial system development organization manage with available requirements engineering processes? This is hard to know as requirements engineering research often falls short in characterizing the scalability of proposed methods.

3.4 Requirements Engineering for Requirements Engineering

Joerg Doerr (Fraunhofer IESE – Kaiserslautern, DE)

Joint work of Doerr, Joerg; Hess, Anne
Main reference A. Gross, J.Doerr, "What You Need Is What You Get! The Vision of View-Based Requirements Specifications," in Proc. of 20th IEEE Int'l Requirements Engineering Conf. (RE), pp. 171-180, 2012.
URL http://dx.doi.org/10.1109/RE.2012.6345801

Software requirements specifications play a crucial role in software development projects. Especially in large projects, these specifications serve as a source of communication and information for a variety of roles involved in downstream activities like architecture, design, and testing. The Dagstuhl Workshop claims that there is a major shift in how we need to approach the RE task due to changes in computational paradigms and to development of organizational capabilities. This position paper argues that our RE community has currently little knowledge about what our stakeholders, i.e., the stakeholders of requirements specifications want to see in requirements specifications and that this will even be worse in the light of the prospected major shift to RE. It argues that in order to create high-quality requirements specifications that fit the specific demands of successive document stakeholders, our research community needs to better understand the particular information needs of requirements specification's stakeholders, but especially the downstream development roles like architects and testers. So more Requirements Engineering for Requirements Engineering needs to take place.

3.5 Requirements Management for Service Providers: the Case of Services for Citizens

Xavier Franch (UPC – Barcelona Tech – Barcelona, ES)

Main reference X. Franch, "Requirements Management for Service Providers: the Case of Services for Citizens," arXiv:1301.4600v1 [cs.SE], 2013.
URL http://arxiv.org/abs/1301.4600

Take the Internet of Things, a piece of cloud computing, a handful of smart cities, don't forget social platforms, flavor it with mobile technologies and ever-changing environments, shake it up and... voilà! What a wonderful service! Oops! Wait a minute, where did my requirements go?

3.6 The Importance of Continuous Value Based Project Management in the Context of Requirements Engineering

Gilbert Fridgen (Universität Augsburg, DE) and Julia Heidemann (McKinsey & Company, DE / Universiät Regensburg, DE)

Joint work of Fridgen, Gilbert; Heidemann, Julia
Main reference G. Fridgen, J. Heidemann, "The Importance of Continuous Value Based Project Management in the Context of Requirements Engineering," arXiv:1301.5438v1 [cs.SE], 2013.
URL http://arxiv.org/abs/1301.5438

Despite several scientific achievements in the last years, there are still a lot of IT projects that fail. Researchers found that one out of five IT-projects run out of time, budget or value. Major reasons for this failure are unexpected economic risk factors that emerge during the runtime of projects. In order to be able to identify emerging risks early and to counteract reasonably, financial methods for a continuous IT-project-steering are necessary, which as of today to the best of our knowledge are missing within scientific literature.

3.7 Requirements Engineering Discovery in Open Source Software Projects

Anna Hannemann (RWTH Aachen, DE)

Open source software (OSS) presents a class of successful community-driven systems. Software engineering (SE) in OSS is a subject of many studies. The researchers discover, investigate, and even simulate the organization of development processes within open-source communities using data from publicly available OSS code repositories and communication platforms. However, organization of requirements engineering (RE) in OSS is seldom addressed. The requirements in OSS are not explicitly defined. They are intertwined into the development and communication process of open-source community members.

By analyzing RE in OSS, we can learn, how different layers within communities are integrated in OSS development process. Does their voice matter? Which structural and organizational changes do influence significantly an OSS project in general and organization of RE in particular? Are there different participation model in terms of requirements negotiation? The answers to these and other related research questions can help us to design community-oriented RE for development of complex, socio-technical systems within interdisciplinary, distributed teams.

3.8 Requirements Engineering as a Distributed Cognitive Process

Sean Hansen (Rochester Institute of Technology, US)

Effective requirements engineering has remained a persistent impediment to the success of information systems projects. In this research, we undertake a novel reframing of requirements engineering as a socio-technical distributed cognitive process in which diverse stakeholders collaborate to reach a collectively-held and feasible understanding of design requirements. This pursuit of shared understanding represents the 'computing' of a closure on a requirements set based on distributed representations. We draw upon the theory of distributed cognition to analyze the ways in which requirements processes become distributed across social, structural, and temporal boundaries and how the requirements computation unfolds in diverse development environments. In this position paper, we highlight the complementarities and challenges that a distributed cognitive perspective presents for long-standing topics in requirements engineering research. In addition, we posit a number of new lines of inquiry that this approach engenders.

3.9 Orthogonal Perspectives on Taming Complexity

Jane Cleland-Huang (DePaul University – Chicago, US)

This position paper discusses several different forms of software complexity. It describes a variety of techniques for managing, embracing, and living-with complexity throughout the software development life-cycle. Finally, the paper advocates creating strategic traceability links that can be used to generate comprehensible and meaningful views that slice the system and provide the stakeholders with the information they need to perform specific tasks.

3.10 Walk Before You Run: A Dialogue with Three US Developers on "Within" Complexity of Requirements

Jane Huffmann Hayes (University of Kentucky, US)

When considering how to manage complexity within requirements, it is prudent to understand industry opinion. Toward that end, three developers were consulted, one from IBM, one from HP (a startup bought by HP), and one from a small industrial programming company (use some automation to accomplish manufacturing). Based on their feedback, it is the position of this author that much of Industry is still attempting to ensure capture and use of requirements; we may be asking them to run before they walk if we 'push' techniques aimed at managing complexity. The questions and industry responses follow. A suggestion of a first step toward addressing complexity closes out the paper.

3.11 Complexity Explained

John Leslie King (University of Michigan – Ann Arbor, US)

The title for this was assigned to me. As my colleague said when the assignment was made, “I’m sure you can talk about this.” Hmmm. Turns out he was right: I can talk about it. And it is important.

People use the term “complexity” a lot in requirements, but there is more to it than meets the eye. What people often talk about as complex is, in fact, complicated. What is the difference? Complicated means lots of parts, and perhaps lots of causal connections between the parts. Complicated might be hard to understand but it can be understood. Complex, in contrast, typically means things that are changing, evolving, becoming something that is at once similar to and different from the thing it used to be. The important issue is control. We typically at least can control things that are complicated; we typically cannot control things that are complex. This does not mean that we cannot understand complex things, we sometimes can. But it is hubris to think that just because we understand something we can control it. And it is worse than hubris to think we can control things we do not understand.*

This brings us to the issue of requirements. I used to think that requirements engineering was a way to help us address complexity. I have changed my mind on that. I think that requirements engineering can be useful when helping us to create systems that address complicated problems, but the moment we cross from complicated into complex, requirements engineering fails us. Put simply, we can build software that deals with complicated things, but we cannot build software that deals with complex things. And requirements engineering – or any other methods-based approaches to improving software development – will save us from this fact. The smart requirements engineer, therefore, keeps the effort focused on dealing with the complicated, and out of the complex. Complicated is hard enough. Complex is impossible. Maybe this should be a new tenet of requirements engineering: keep it complicated, but not complex.

---

*We deal all the time, and often successfully, with things we do not understand at some level. We have managed to control many kinds of physical materials without fully understanding the nature of matter. We have applied ‘fuzzy logic’ to making things work even though we do not completely understand them. Good workarounds allow us to work around problems. Also, what we can control changes over time, as we come to understand things or create workarounds.

3.12 Where is the human mind in requirements engineering (research) and what do we think and know of it?

Kim Lauenroth (adesso AG – Dortmund, DE)

We have to reconsider our theories, methods, models and tools taking into account the limitations and abilities of the human mind on both sides, the requirements engineer’s side and the stakeholder's side. Without a profound understanding of our own mind, the development of requirements engineering theories, methods, models, techniques etc. is not optimal since we are developing 'software' for a machine that have not understood properly.

3.13 Software Requirements Are Soft

Julio Cesar Leite (PUC-Rio de Janeiro, BR)

Software requirements and the engineering of software requirements stands upon the name require and the suffix: ment, which according to the Online Etymology Dictionary means: "suffix forming nouns, originally from French and representing L. -mentum, which was added to verb stems sometimes to represent the result or product of the action.". It occurs that the result of the action require, in this context, is a fuzzy one. As such, the construction of software suffers from a basic hard problem, an unstable soil. Dealing with environment instability is hard. We are studying the concept of requirements awareness as a way of tackling environment instability in order to support software evolution policies.

3.14 The evolution of requirements: towards an ecological theory

Kalle Lyytinen (Case Western Reserve University – Cleveland, US)

This paper addresses long waves and changes in the requirements engineering environments similar to punctuated shifts studied in organization theory or Kondratieff waves in macroeconomics. The is issue at stake is are the tectonic shifts how the requirement work is expected to conducted. My answer is yes. The key is that requirements engineers face a larger context, a larger multitude of mechanisms and new dynamics in which design spaces and solution spaces interact - the key target of requirements work. While the old world of requirements engineering was largely engaged in digitizing the cowpaths and where the movement was from existing task or function to a isolated computer solution. Here the key challenges were cognitive, economic (cost/risk) and technological and requirements were carried out to estimate the failure to execute the plan. In the new world the design and solution spaces are recursively and dynamically organized and involve a constant orchestration of largely existing digital assets to come to a wholly new computational solution which does not have counterpart in the past. The key question to ask is : "What will you do when you can compute anything?" In such situations it is assumed that the effect size of technology increases as variation in combining technological assets increases (Arthur 2010). Overall, this means that the mutation rate of technologies and solutions change resulting in shifts in ecologies where solutions and problems are matched. The key question what requirements engineers must address is to find applications/system that match technological capability and stakeholder needs often in radical/disruptive manner. This is a generative model which assumed multiple evolutionary paths for software and system evolution through functions of informating, embedding into new contexts, and expansion of technological capabilities. The key drivers for variation are classic forms of evolutionary change: code and task mutation, selection, retention mechanisms. The key form of RE is Exploration where analysts seeks to minimize the risk of failing to discover.

The main challenges are entrenchment and related cognitive barriers, speed at generating variety which requires experimentation, understanding the effects of asset ecologies and forms of platformization (architectural control), and the effects of network externalities on software asset use. I offered an example from the evolution of music industry and related digital assets how new software and service solutions have emerged over the last 20 years.

3.15 Interactive Traceability Querying and Visualization for Coping With Development Complexity

Patrick Maeder (TU Ilmenau, DE)

Main reference P. Mäder, "Interactive Traceability Querying and Visualization for Coping With Development Complexity," arXiv:1301.5363v1 [cs.SE], 2013.
URL http://arxiv.org/abs/1301.5363

Requirements traceability can in principle support stakeholders coping with rising development complexity. However, studies showed that practitioners rarely use available traceability information after its initial creation. In the position paper for the Dagstuhl seminar 1242, we argued that a more integrated approach allowing interactive traceability queries and context-specific traceability visualizations is needed to let practitioner access and use valuable traceability information. The information retrieved via traceability can be very specific to a current task of a stakeholder, abstracting from everything that is not required to solve the task.

3.16 Requirements Complexity and Evolution: A Computational Perspective

John Mylopoulos (University of Toronto, CA)

Joint work of Ernst, Neil; Borgida, Alexander; Mylopoulos, John; Jureta, Ivan
Main reference E. Neil, A. Borgida J. Mylopoulos, I. Jureta, "Agile Requirements Evolution via Paraconsistent Reasoning", in Proc. of 24th Int'l Conf. on Advanced Information Systems Engineering (CAiSE'12), Gdansk, LNCS, Vol. 7328, pp. 382–397, Springer, 2012.
URL http://dx.doi.org/10.1007/978-3-642-31095-9_25

We review the requirements problem as defined by Jackson and Zave [2]. We then discuss how computational complexity creeps in and how to cope with it. In addition, we sketch some approaches for dealing with requirements evolution, adopted from the PhD thesis of Neil Ernst [1].

References

1 N. Ernst, A. Borgida, J. Mylopoulos, I. Jureta (2012) Agile Requirements Evolution via Paraconsistent Reasoning, Proc. 24th Int. Conference on Advanced Information Systems Engineering (CAiSE'12), Gdansk, June 2012.
2 M. Jackson, P. Zave (1995). Deriving specifications from requirements: an example. Proc. 17th ICSE.

3.17 Large Scale Business System Evolution

Andreas Oberweis (KIT – Karlsruhe Institute of Technology, DE)

Business systems are sociotechnical systems, which are embedded in an organization. The organization itself is embedded in supplier, customer and/or technical markets. Business systems usually contribute to an organization's (strategic) goal.

In the literature several definitions for the term "evolution" have been proposed, sometimes inspired by the interpretation of this term in biology. In practice of business systems the term evolution can be found in different variants: evolution as the result of a proactive plan for sequences of change, evolution as a sequence of some spontaneous or random change events, and evolution as reactions to environmental changes.

Business systems evolve in evolution cycles (or life cycles). However, the business system evolution cycle is strongly related to the evolution cycles of business process, business objects, organizational structures, markets, and legal systems. A business system consists of components, each one possibly having an evolution cycle of its own.

Business systems are individually implemented or results of customizing a standard software system. In case of standard software systems there are overlapping evolution cycles of the standard business system (under responsibility of the software vendor) and of the customized business system (under responsibility of the software buyer). Some important challenges of complex business system evolution are:

- Develop mechanisms to support synchronization of different evolution cycles (the term "synchronization" still has to be clarified).
- Find optimization criteria for synchronization efforts (besides cost).
- Decide between central and decentral control of evolution (if possible).
- Manage (reduce?) complexity of relationships between different evolution cycles.
- Decide between system replacement and system evolution. Can an evolving business system finally die?
- Develop languages to model and methods to analyze system evolution and evolution cycle synchronization.

3.18 Requirements Engineering Intelligence: Dealing with Complexity and Change

Barbara Paech (Universität Heidelberg, DE)

Similarly to business intelligence, requirements engineering intelligence captures data about software and its development, operation and use, and leverages intelligent mechanisms such as mining and analytics for decision support in requirements engineering and management. In this paper we distinguish usage, system and operation, and development process data capturing either plan, rationale or execution. We discuss open issues in answering the following questions based on this data: what is the current situation of the system and its usage,what changes to the system are necessary and why, what is the impact of a change, when should which change be executed.

3.19 Managing Requirements Evolution in Software Product Lines

Xin Peng (Fudan University – Shanghai, CN)

A software product line (SPL) is a set of similar applications that share a common, managed set of features and are developed from a common set of core assets. A produce line evolves when existing applications change, core assets evolve, or new applications are derived. In an ideal setting, applications always keep consistent with the core assets in the evolution of a product line. However, in real product lines, it is often the case that applications evolve independently to some extent and may deviate from core assets. Therefore, for SPL evolution management, there is a need to balance independent application evolution for quick response and the desire of SPL evolving as a whole. To this end, SPL evolution management needs to monitor and track the evolution trends of core assets and applications and conduct periodic synchronization among them.

3.20 Software evolution in complex environments

Barbara Pernici (Politecnico di Milano, IT)

Evolution of software can be considered from several different points of view. First of all, evolution can be an issue in service-based systems, where services adapt themselves on the basis of a set of available adaptation actions. For instance, service compositions can select different components, or Quality of Service requirements for invoked service may vary.

A wider perspective is given when complex environments are considered, in which external factors are involved, as in the case of software running in variable contexts or changing users' requirements.

Project such as the S-Cube (the European Network of Excellence in Software Services and Systems, http://www.s-cube-network.eu/) have studied the impact on the software life cycle of adaptive environments, where services adapt to changes. The life cycle in this case is composed of two cycles, one focusing on design for adaptation, the other one on adaptation at execution time, where monitoring is the basis for identifying contexts which require adaptation to satisfy requirements in a changing environment. Adaptation includes strategies such as dynamic service selection, substitution, repair, and so on. When available adaptation strategies are not sufficient to meet the requirements, a redesign of the application can be considered, thus evolving the existing system.

However, one of the goals in adaptive systems is to minimize this evolution and to perform it only when necessary, minimizing change in the system, As a consequence, research goals are to study the areas of designing applications for adaptation, trying to improve their dependability, to minimize evolution, and to identify adaptation contexts and patterns.

On the other hand, evolution has often to be analyzed in the context of complex environments, including not only software components, but also users and business level requirements and the underlying physical infrastructure. In this case, evolution is based on monitoring and control at different levels, and several factors need to be taken into account:

- considering multiple instances and multiple applications

- goals might be variable and context-dependent
- the effect of adaptation decisions may be not always clear.

As an example, adaptation of services and systems with the goal of reaching energy efficiency and optimizing the environmental impact of IT (e.g., as considered in the European projects GAMES – http://www.green-datacenters.eu/ – and ECO2Clouds – http://eco2clouds.eu), requires to consider multiple layers at the same time: applications, services, and infrastructure layers.

In this case many conflicting goals might be present, and actions intended to reach a set of goals might have a negative impact on other ones.

Therefore, when considering complex environments, evolution should take into consideration different aspects:

- rather than considering single applications, patterns of usage of applications should be taken into account
- user behavior can change, also considering feedback from the monitoring system
- the definition of requirements in a multi-layer and adaptive system needs a redefinition of the way requirements are usually specified and considered
- in addition to requirements on functionalities and quality of service, data usage requirements should also be considered.

From the discussion in the workshop, ideas emerged on dealing with manageable evolution (distinguishing between complicated and complex systems) and the need of managing requirements at different abstraction levels.

3.21 Boundary Spanning in RE

Balasubramaniam Ramesh (Georgia State University, US)

Joint work of Ramesh, Balasubramaniam; Mohan, Kannan

Boundary spanning is a central activity in requirements engineering. Research on viewpoint management, requirements traceability and development methods provide several mechanisms that facilitate boundary spanning by suggesting boundary objects and boundary spanning roles. Our research examines the role of boundary spanning in three different contexts: collaborative development, projects characterized by cognitive conflicts and mixed motive conflicts. Our findings suggest that boundary spanning practices must be tailored to the specific project context. In addition, the evolving nature of boundaries due to changes in the external, organizational, and project context necessitates adaptations/evolution of RE tools/processes that fit the context.

3.22 Dealing with uncertainty and iterations in design processes: An entrepreneurial perspective

Isabelle Reymen (TU Eindhoven, NL)

This abstract aims to question the relation between requirements management and two important characteristics of design processes, namely the iterative character and dealing with uncertainty about the environment. It brings forward the entrepreneurial effectuation theory [6] as a perspective that might inform requirements management.

An important characteristic of design processes is iteration [4]. Much of the complexity of design processes is due to iterations [10]. A recent study focusing on long-term process dynamics in design processes in a real-life context [1] found that the iterativity of designing occurs along multiple dimensions, at multiple levels of analysis, and on multiple different time scales. Another important characteristic of design processes is dealing with uncertainty about the environment. Simon [9] already noted the importance of dealing with changing complexities in the outer environment.

Effectuation theory is a recent entrepreneurship theory developed by Sarasvathy [6]. The creation of new ventures is a process characterized by the need to decide and take action in the face of uncertainty. The entrepreneurship literature to date has advanced two contrasting approaches to decision-making (so-called decision-making logics) under uncertainty. The first logic aims to lower uncertainty based on prediction and is termed "causation" by Sarasvathy [6]. Cuastion is contrasted with a second logic labeled "effectuation". Effectuation allows to control uncertainty by being adaptive, seeking feedback and leveraging existing means and stakeholder contacts. The most important principles behind the effectuation logic are means orientation (instead of goal orientation), affordable loss (instead of expected return), strategic alliances (instead of competitive analyses) and exploiting contingencies (instead of preexisting knowledge).

Effectual logic is thus especially useful for (design) decision making under situations of high uncertainty, leaving also room for much iteration in the design process. What can we learn from this decision making logic for managing requirements in complex design processes? It would be worth trying to apply the effectuation principles for designing complex processes. Agile methods, and especially the agile design method Scrum [8] comes really close to the effectuation principles and are already used in (software) design processes in practice. Evaluation of these processes from an effectuation perspective can give insight in the pros and cons of applying effectual principles in design practice, more specifically, under which conditions does they work, and which type of iterations are supported in the process.

When finding inspiration for the requirements process in using effectuation logic, it is important to realize that recent studies point at the combined use of effectuation and causation in practice. Whereas effectual and causal logics were often contrasted in the literature, there is some empirical evidence that individual entrepreneurs use both logics [7], [3]. The combination of effectuation and causation was also found empirically in innovation/design processes in small firms [2]. In addition, Sarasvathy suggests that ventures should switch between effectual and causal logics depending on the degree of uncertainty the venture is confronted with. A recent study [5] focuses on the potential shifts in effectual and causal decision-making over time (in new venture development processes). Such a dynamic perspective is necessary for moving beyond the discussion of causation and effectuation as contrasting approaches. It can shed light on whether, how and why both approaches are alternated or combined over time. These insights can then also be applied to the requirements managing field, ultimately retrieving guidelines for how to organize requirements processes taking into account increasing complexity and allowing for several type of iterations in the design process.

Concluding, ingredients for finding a relation between requirements management and two important characteristics of design processes, namely the iterative character and dealing with uncertainty about the environment are discussed, thereby making use of the entrepreneurial effectuation theory; but the relations are not yet clear. I hope at least some food for thought was offered that initiates nice discussions on these topics.

References

1 Berends, H., Reymen, I.M.M.J., Stultiens, R.G.L., Peutz, M. (2011) External designers in product design processes of small manufacturing firms. Design Studies, 32(1), pp. 86-108.

2 Berends, H., Jelinek, M., Reymen, I.M.M.J., Stultiens, R.G.L. (2012) Product Innovation Processes in Small Firms: Combining entrepreneurial effectuation and managerial causation, Journal of Product Innovation Management. (forthcoming)

3 Dew, N., Read, S., Sarasvathy, S.D., Wiltbank, R. (2009) Effectual versus predictive logics in entrepreneurial decision-making: Differences between experts and novices. Journal of Business Venturing, 24: 287-309.

4 Dorst, K., & Cross, N. (2001) Creativity in the design process: co-evolution of problem-solution. Design Studies 22, 425-437.

5 Reymen, I.M.M.J., Andries, P., Berends, H., Mauer, R., Stephan, U., van Burg, J.C. (2012) Dynamics of Effectuation and Causation in Technology- based New Ventures, Proceedings of the 2012 Academy of Management Annual Meeting, August 3-7, 2012, Boston, Massachusetts. (pp. 1-39). Boston: Academy of Management.

6 Sarasvathy, S.D.(2001) Causation and effectuation: Toward a theoretical shift from economic inevitability to entrepreneurial contingency, Academy of Management Review, 26(2), pp. 243-263.

7 Sarasvathy, S.D. (2008) Effectuation: Elements of entrepreneurial expertise. Northampton, MA: Edward Elgar.

8 Schwaber, K. and Beedle, M. (2002) Agile Software Development with SCRUM, Upper Saddle River, NJ, Prentice-Hall.

9 Simon, H.A. (1996) The sciences of the artificial, 3rd ed, MIT Press, Cambridge, Massachusetts.

10 Smith, R. P., & Morrow, J.A. (1999) Product development process modelling. Design Studies 20, 237-261.

3.23 Understanding Software System Evolution through Requirements Monitoring

William N. Robinson (Georgia State University, US)

Main reference W.N. Robinson, S. Fickas, "Designs Can Talk: A Case of Feedback for Design Evolution in Assistive Technology," in Design Requirements Engineering: A Ten-Year Perspective, K. Lyytinen, P. Loucopoulos, J. Mylopoulos, and W. Robinson, Eds., pp. 215–237, LNBIP, Vol. 14, Springer-Verlag, 2009.
URL http://dx.doi.org/10.1007/978-3-540-92966-6_12

This paper presents the challenge of understanding system behaviors. Software systems are complex, and we do need the traditional computing techniques, such as requirements discovery, modeling, simulation, and automation. Improvements in these traditional areas will significantly improve software and software development. New approaches may also be beneficial. Herein, we consider the relatively new approach of requirements systems monitoring (not to be confused with traditional techniques of systems monitoring or telemetry). Requirements monitoring is fundamentally an interpretation problem: What is the system doing?, as expressed in high-level behaviors and qualities. This perspective can be applied to software execution or to software development, where the system is the community of developers. In both contexts, we want to know if the system is meeting its requirements and moving toward a successful conclusion. Monitoring includes specific techniques, but also includes the philosophy that continuous monitoring enables continuous improvement, which is fundamental to successful systems in uncertain and evolving environments.

3.24 Getting to the Shalls: Facilitating Sense-Making in Requirements Engineering

Christoph Rosenkranz (Goethe-Universität Frankfurt am Main, DE)

Knowledge transfer, communication, and shared understanding between project stakeholders are critical, problematic factors in requirements engineering (RE). Yet, specifying complete and unambiguous requirements in the face of the complexity inherent in RE remains a significant challenge. There is a lack of systematic procedures that facilitate a structured analysis of the qualitative data in RE. We propose the use of a procedural approach to fill this gap that builds on Linguistic Analysis and Grounded Theory Method.

3.25 Platforms wars as a source of complexity

Matti Rossi (Aalto University, FI)

For last 15 years most end-user applications have been developed for Windows and server software for various flavors of UNIX. There were other platforms, but they were niches. This made certain platform constraints easy to deal with and more or less fixed.

All this has changed in last 5 years. A proliferation of mobile platforms and totally new user interaction forms has created a maze of options to work with. It would be easy if everyone just developed software and services using HTML5 as a standard rendering platform and browser as an execution platform. However, the major operating systems are shifting and it is not Chrome the browser that is the largest operating system, but rather Android that competes with iOS. This creates a wealth of issues.

First issue to choose is whether to develop a user interface for desktop, mobile or tablet form factor. Then one has to decide whether to use mouse, touch, menus etc. All this and still the actual execution platform and its exact form factor and API's have to be selected. When these are done, one can choose a distribution channel (store) and perhaps an in-app payment method. If the choices are right and the developers have luck and good timing, they have an instant potential and addressable market of tens or even hundreds of millions devices and users. However, at the same time the platforms are on the mercy of fashion and fads. So if wrong constraints are acknowledged in the start and/or development is delayed, it could be that a new killer app is deployed against last season's platform, which is no longer viable.

To understand the platform risks and constraints, there should be more research on path dependencies and whether we are moving towards hardware or software platforms. Furthermore, the economics of moving to fragmented and closed versus open platforms should be studied to understand the long term evolution.

3.26 Extreme Requirements: The Challenge of Ultra Large Scale Systems

Alistair G. Sutcliffe (Univ. of Manchester, GB)

I review the drivers of complexity in very large scale systems arguing that human unpredictability, exception conditions in social systems, functional bloat, environmental change, and increasing connectivity are inescapable trends creating over complex, global scale systems which current RE methods can not address. Dealing with complex systems is confounded by long development times in changing worlds and communication problem within groups of developers. Possible solutions for large scale socio-technical systems are explored, ranging from simulations to understanding the constraints on high level system goals, to evolutionary approaches which generate and test solutions, where requirements become fitness criteria for surviving in operational environments. Finally I propose a challenge for developing a sound abstraction theory to bridge requirements to conceptual system architectures and argue that abstraction theory will be necessary to effectively 'divide and conquer' requirements problems in complex systems.

3.27 A general-to-specific architectural design for managing requirements evolution

Fan Yang-Turner (University of Leeds, GB)

Deriving requirements from users to support issues with cognitive complexity (such as sense making or creative thinking) or social complexity (such as trust enhancement or culture awareness) are difficult. Applications that support these complex issues are often developed in a fast prototyping manner. The requirements are constantly evolving because of the interaction between the users and designers catalyzed by the prototype. Meanwhile, few theories or conceptual models from cognitive science or social science have been reused to the level of design or development. To manage requirements evolution for cognitive or social complex issues, I posit a general-to-specific architectural design underpinned by theoretical models. In this paper, I illustrate this design in a development project, which creates a platform that facilitates sense making and decision making in data-intensive and cognitively-complex settings.

3.28 Modeling and (social) complexity – a perspective from conceptualizing the BI-enabled adaptive enterprise

Eric S. Yu (University of Toronto, CA)

Is complexity good? Is complexity bad? Modern living takes for granted numerous conveniences that are built upon layers upon layers of complexity, most of which are hidden from us.

Complexity is good, as long as someone else takes care of it. Complex societies derive value from complexity. We are all benefitting from complexity. Social complexity is achieved through mechanisms of social organization, such as division of labour, specialization, trust, commitment, delegation, exchange, markets, power, and politics. Complexity is hidden by exploiting locality, but is exposed when the locality boundary fails, resulting in (side-)effects propagating across boundaries.

Requirements arise at the boundary between localized social units (abstract social actors), as dependencies from service consumer to service provider. Requirements are inherently distributed. There are numerous consumer/provider interfaces in a society or in an enterprise. In the classical requirements paradigm, the focus is on the one-time activity of obtaining a set of high-quality requirements for a single "system". This paradigm applies only in the context of inflexible, slow-moving, and monolithic service providers. This no longer works in the "brave new world" [1].

Requirements dynamics result from dynamics in the service consumer and in the service provider. Since consumer and provider each evolves at its own pace, driven by different forces and varying degrees of uncertainty, misalignment can be expected between what is wanted by the consumer and what is offered by the provider. Perfect alignment, or perfect achievement of requirements despite ongoing change, is unrealistic in the brave new world. Instead, requirements complexity should be addressed by architecting the enterprise (or society, or relevant ecosystems) so as to minimize misalignment at the boundaries between the various social units, taking into account the abilities (and limits) that each unit has in adapting to its environment.

This perspective on requirements derives from a project on "BI-enabled Adaptive Enterprise Architecture", a project within the Business Intelligence Network (BIN), a Canadian academic-industry collaborative research network. The project aims to position BI and data analytics within a conception of the adaptive enterprise [2].

References

1 M. Jarke, P. Loucopoulos, K. Lyytinen, J. Mylopoulos, and W. Robinson (2011). The brave new world of design requirements. Inf. Syst. 36, 7 (November 2011), 992-1008. http://dx.doi.org/10.1016/j.is.2011.04.003

2 E. Yu, S. Deng and D. Sasmal (2012). Enterprise Architecture for the Adaptive Enterprise. Proc. 7th Workshop on Trends in enterprise architecture research (TEAR). at The Open Group Conference 2012, Barcelona, Spain, October 23–24, 2012. http://dx.doi.org/10.1007/978-3-642-34163-2_9

3.29 A Feature Model Centric Approach to Requirements Management and Reuse

Haiyan Zhao (Peking University, CN)

As a critical component of software reuse, requirements reuse provides organizations with the ability to share requirements across projects without absorbing unnecessary duplication of artifacts. The key issue in requirements reuse is *how to make the requirements reusable*. Requirements are reusable only if they can be easily configurable with respect to the needs and preference of users in the sense that the requirements should be highly customizable following the principles of loose-coupling and high-cohesion. To this end, this talk proposes and discusses an integrated framework for managing and reusing requirements, which adopts feature models as the centric representations to organize and manage the reusable requirements. It encompass 1) the structures of feature models and the mechanism for requirements family evolution; 2) the construction of traceability between feature models and other artifacts, such as use cases and software architecture; 3) the reuse of requirements through feature model customization.

3.30 Requirements Issues in SoC and SoS

Andrea Zisman (City University – London, GB)

Main reference A. Zisman, "Requirements Issues in SoC and SoS," arXiv:1301.5470v1 [cs.SE], 2013.
URL http://arxiv.org/abs/1301.5470

This position paper "Requirements Issues in SoC and SoS" outlines some of the existing issues and challenges concerned with the complexity of the requirements for service-oriented computing and systems-of-systems applications.

3.31 A quest for building theories of requirements evolution

Didar Zowghi (Univ. of Technology – Sydney, AU)

Main reference V. Gervasi, D. Zowghi, "On the Role of Ambiguity in RE," in Proc. of the 16th Int'l Working Conf. on Requirements Engineering, Foundations for Software Quality (REFSQ'10), LNCS, Vol. 6182, pp. 248–254, 2010.
URL http://dx.doi.org/10.1007/978-3-642-14192-8_22

The primary productive force in information capitalism is the human mind and its cognitive, linguistic, and creative capacities. New systems of production and exchange, and new orders of power and control are built to harness, enclose, and exploit what humans can do naturally: think, communicate, problem-solve, and create etc. Software systems are unique artefacts both produced by, and helping to produce, informational capitalism. Similar to the larger systems of power in which software is situated and embedded, it can often be unstable, uncertain, unpredictable, and prone to disruption and failure. Yet paradoxically, each new software application developed carries the hopes of human users for order and certainty, offering a technological solution that will effectively solve a specific problem or support a specialized activity or process. No matter how often software systems might under-perform or fail given the weight of collective and sometimes unrealistic expectations, software users remain optimistic that 'this time it will be different!'

What role should requirements engineering research and practice play in increasing the confidence of users of future systems that their voices will be heard and their needs will be met better than it has been over 6 decades of software systems development? More importantly, how could effective RE practices decrease the likelihood of future system failures, disruptions, and disorder in everyday life? What we as a community have achieved in RE research thus far that has produced a quantum leap in that direction? These are some of the questions that many of us would like answered.

I have been exploring some old and recent research questions and issues (summarized below) that I hope may contribute to responding to some of the above questions.

User involvement in RE and software development

On the one hand, software development processes often focus on a small number of representative users to give input and feedback in requirements related activities and later during user acceptance testing. On the other hand, user feedback mechanisms in software systems are not standardized and remain rather ad hoc. Typically base line features of software approved by management are perceived to be more important than those requested or needed by less frequent or less powerful users. Communication channels that allow for effective collaboration among users or between users and developers are usually decoupled from software systems and their development infrastructure. Research involves conducting field studies of software development and software usage, so that we can collect valuable data through observation of software development tasks and interactions through interviews with as many relevant stakeholders as possible within the entire software development lifecycle.

The aim is to employ Grounded Theory throughout this longitudinal research project in order to increase our understanding of the evolution of requirement knowledge generated by users involvement. We are specifically interested in building theories of generation, negotiation, validation and sustainability of requirements knowledge through the lens of users participation. From a management perspective, we wish to develop a sufficiently deep analysis of the evolution of the role of users involvement in software development and how this is managed by development teams.

Properties of Requirements Specifications (RS) are attributes that can be associated with the documented requirements. Properties are important because they enable us to assess whether an RS is in some way "good" or "better" than another RS. Knowing this, one can decide when to declare the requirements process complete, how to estimate the cost of a development effort, and so on. I have studied factors such as inconsistency, incompleteness, and ambiguity and investigated their evolution and management within the RE process. An individual cannot inspect or examine a requirements specification and determine its level of ambiguity, consistency, completeness, etc., without referring to external bodies of knowledge such as other related documents, business processes, organizational strategy, other systems, and so on. Many external bodies are highly volatile while others are slow to change while others are completely static. There is a need to develop sound theories that describe the dependencies between RS properties, requirements and external bodies of knowledge, and show how a number of common phenomena (and related properties) can be expressed and modeled by that theory. In particular, modeling the interdependencies and interrelationships between these attributes that determine the overall quality of the RS. Here I mention three of such properties:

Consistency
Every phenomenon that has its ultimate roots in human subjectivity is liable to possible change in the future. In practice it often happens that revision of requirements comes either from strikingly new and different needs which demand that the conceptual model be revised in order to account for them, or from some unexpected conclusions which are deduced from the conceptual model itself and which may contradict some of the already known requirements [2]. The problem of managing changing requirements and maintaining the consistency of evolving requirements throughout software development life cycle is one of the most significant issues in requirements engineering.

Completeness
The boundary between the real world and the application domain is necessarily a leaky interface because the individuals and predicates continue to be discovered and captured during requirements evolution and hence this boundary is volatile. Therefore, at any point in the evolution of a system, one cannot claim with absolute certainty that all the items of interest and their relationships have been captured completely, because the application domain itself (where the requirements are situated), is indeed an evolving entity. This argument suggests that absolute completeness of requirements specifications can never be established and completeness remains only as a relative measure [2], [3].

Ambiguity
Ambiguity has long been considered as one of the worst defects in requirements specifications whose identification and removal should be given higher priority. More recently the nature of ambiguity has been subject of research that advocates that the simplistic view of ambiguity as merely a “defect” that has to be avoided at all costs does not do justice to the complexity of this phenomenon [1].

References

1 Gervasi, V. and Zowghi D., (2010), On the Role of Ambiguity in RE, Proceedings of the 16th International Working Conference on Requirements Engineering, Foundations for Software Quality (REFSQ’10), Essen, Germany.
2 Zowghi D. and Gervasi V., (2002), The Three Cs of Requirements: Consistency, Completeness and Correctness, proceedings of 8th International Workshop on Requirements Engineering: Foundation for Software Quality, (REFSQ’02), Essen, Germany.
3 Zowghi D., and Gervasi V., (2004), On the Interplay Between Consistency, Completeness, and Correctness in Requirements Evolution, Journal of Information and Software Technology, 46(11):763-779.

4 Working Groups

4.1 Managing Complexity within Requirements

Andrea Zisman (City University – London, GB)

Joint work of Zisman, Andrea; Maeder, Patrick; Hayes, Jane; King, John; Lauenroth, Kim; Mylopoulos, John; Ramesh, Bala

In this working group we discussed the main topic of how to manage complexity within requirements. We started with some initial presentations from the members of the group about (i) cognition effects of the human mind and how these effects can influence requirements engineering activities; (ii) industrial experience with requirements; (iii) difference between complicated and complex; (iii) use of traceability queries and visualization; and (iv) requirements issues when developing SoC and SoS. We also discussed what could be the causes of complexity, and how to manage complexity within requirements. Some of the ideas that have been proposed to manage complexity within requirements are: (a) identification of dependencies and opportunities in requirements, (b) use of test-driven development, (c) use structural levels of constructions, (d) use of specific terminology, and (e) keep the project out of the complex space and contain its scope.

4.2 Requirements Discovery and Negotiation in Complex Environments: Work Group Discussion

Gilbert Fridgen (Universität Augsburg, DE)

Joint work of Fridgen, Gilbert; Hansen, Sean W.; Reymen, Isabelle M.M.J.; Rosenkranz, Christoph; Svensson, Richard Berntsson; Yu, Eric S.; Zowghi, Didar

During the discussion we structured the topic in four areas that have to be considered for requirements discovery and negotiation in complex environments: The role of humans, the design attitude, the difficulties that come with complexity, and the challenges through evolution.As for human beings, we first have to distinguish between software engineers who have to deal with complexity during development and end-users who have to cope with a possible insufficient system. Both stakeholders can be parts of complex systems of power and politics that can hinder any project, especially if it comes to negotiation. While users are today by far more involved through participatory design (e.g. with agile methods), observed use, misuse, or non-use can be an important indicator for the quality of requirements. To improve communication between software developers and users we have to develop their skill set to enable boundary spanning between disciplinary knowledge. User observation is also an important interlink between to role of human beings and the design attitude. It can be one means of creativity to foster the exploration of user needs that have not been addressed by a requirements specification before. Especially the focus on processes instead of products is important here: Users do not need a certain product for itself; they rather need support in fulfilling their respective processes. To identify and support these processes, software engineers require a design thinking in contrast to a decision oriented thinking, iteratively improving their solutions. This can be reached through multiple cycles of divergence/convergence, effectuation/causation, freezing/unfreezing, and exploration/exploitation.For complexity itself, it is important to consider that it is only a perception by humans and that different individuals can perceive it differently, especially if you differentiate between software engineers and end-users. Modularity seems still to be a powerful means to break down and manage complexity. Successful strategies to overcome this issue are furthermore “markets” for innovative ideas like e.g. implemented by Google who reserve only 70% of the time for work that generates direct revenue, 20% for any project they want and 10% for own idea. It is consequently a means to manage the complexity of many employees having various ideas. Therefore, complexity can be crucial, it can be the source of innovation that needs to be leveraged.Regarding the evolution of systems and their surroundings we have to be aware that we are only partially able to drive evolution and that much evolution will just happen by itself requiring systems to react. We therefore need to identify areas in which we can control evolution and areas in which we have to be prepared e.g. by implementing a certain amount flexibility. The architectural concept of solid cores and flexible boundaries seems to be a promising strategy to prepare systems for a controlled evolution.Out of these challenges we identified to following four research topics to be the most pressing but currently neglected:First, we need to do more shift our research from a product to a process focus. Second, we need to explore the dynamics of power and politics in software development projects to implement corresponding governance structures. Third, we need to improve human-centered discovery techniques, especially including (fourth) improved feedback mechanisms.

4.3 Managing complexity through requirements

Anna Hannemann (RWTH Aachen, DE)

Joint work of Xavier Franch (chair); Hannemann, Anna (scribe); Cleland-Huang, Jane; Doerr, Joerg; Heidemann, Julia; Leite, Julio; Rossi, Matti

The discussion addressed questions along different dimensions: requirements nature (requirements for RE, Jackson's problem frames), techniques (social network analysis), domains (services for citizens, OSS communities) and architectural issues (constraints on mobile technologies, agile vs. traditional architectural model).

In terms of RE organization (e.g., generic vs. OSS projects, services for systems), different iteration cycles can be observed. Different frequency of change requires different approaches to handle the resulting complexity. For example, the complexity of services for mobile systems consists of: too different users, too different context, too different applications, too many sensors. The complexity arising during development and evolution of services can be approached by tailoring the requirement specifications to the development roles and even individuals. First, define the whole picture of the process, roles and tasks. Second, provide every participating engineer with a personalized view of his/her role within the development process. We argue, that the increase of general awareness is one way to improve the understanding of complexity and, thus, to handle it. Social network analysis can be applied in order to make not only technical but also social complexity transparent to the process participants. Socio-technical representation should not only be based on RE related data, but also incorporate communication and interaction aspects within the whole development process.

In terms of reducing the complexity of requirements, prioritizing appears to be an appropriate strategy. The essential (cannot avoid it) complexity can be distinguished from accidental (this comes from the method). The existence of complex requirements may help tackling more complex problems (e.g., constraints to model platform variability) on the one hand. On the other hand, complex requirements are more complex to handle, e.g. to trace. Prioritizing presents a challenging process and can eventually lead to information loss. Especially, considering community-driven systems the open questions to answer are: "Where is the boundary of the system?" and "How to handle the requirements form the long-tail of community".

In terms of RE environments, more prescriptive and modular systems can reduce the complexity of requirements negotiation among multiple stakeholders, different in their views, knowledge and needs. However, knowledge innovation is believed to be a result of a creative, non-restricted process of idea finding. When and why more traditional RE or a generative model is more suitable – an open question.

In terms of complexity of interactions between architectures and requirements, we need to understand what the information needs of our architects are. Then we can elicit and transfer the information to architects. A subset of requirements known as architecturally significant requirements (ASR) drive architectural design, while at the same time existing architectural design constrains new functions (requirements) that the system can implement. This is a cost issue. Almost any new requirements can be accepted, but they will cost more or less according to the necessary refactoring.

4.4 Managing Complex Systems Evolution with Requirements Models

Matthias Jarke (RWTH Aachen and Fraunhofer FIT, DE)

Joint work of Matthias Jarke (scribe), Xin Peng, Barbara Pernici, Alistair Sutcliffe, Hayian Zhao (chair)

The workgroup addressed three major questions from different perspectives:

- Alignment of model types with complex systems and their evolution: models should be aligned to the required and available depth of knowledge; to support evolution, they should be simulatable, but also enable ex-post change through traceability
- Careful management of model incompleteness: requirements scenarios should be modeled at different levels of granularity and timescales, with or without consideration of aspects such as causality or feedback loops; they not only at design time but also at runtime because of constant change; but all these submodels cannot necessarily be fully connected to each other. Especially phenomena models beyond the actual system design are of necessity incomplete, but purpose-focussed traceability patterns may help not to forget critical aspects that someone has thought about before.
- Modeling requirements variation and evolutionary branches: clutter in variation modeling should be avoided using recent solution approaches such as product families even at the RE level; similarly, empirical research should identify process patterns of requirements evolution that can perhaps be standardized.

Alistair Sutcliffe presented Extreme Requirements as the challenge of ultra-large socio-technical systems to illustrate their typical problems: overambition by desire to improve existing systems when transferring them to new settings; the tendency of political lobbying to complicate regulations with endless exceptions; the unpredictability of human users, developers, and especially decision makers; the evolution of technology combined with multiple layers of connectivity and thus sources of additional complexity. He stressed the importance of simulations as well as of not just deterministic but also stochastic goal and obstacle analysis, and the helpful nature of requirements patterns. Hayian Zhao advocated a feature-centric approach to RE as a possible answer to many of these challenges. Features abstract away complexity and evolution in a manner that is relatively easy to communicate among stakeholders. He, as also Matthias Jarke in his brief talk, stressed the role of standard platforms and standard software in combining complexity reduction with richness of innovation. Features are also claimed to provide patterns for traceability, such that Zhao even proposed to replace domain models altogether by feature models. Barbara Pernici focused her talk on the rarely discussed but practically eminently important aspects of evolving systems operation requirements, using the example of high-performance systems in scientific computing. For example, how to make an expensive ocean-simulation environment more energy-efficient, how to re-balance workload between mobile devices and static servers in different phases of battery life. New parameters must be introduced into requirements models to capture these aspects, e.g. frequency of use, data and usage stability, etc.

Finally, Xin Peng focused on the evolution and complexity aspect of socio-technical systems, claiming that traditional systems that separate human and technical components to much are neither social nor technical. In the future, we shall face a deeper integration of human, software, and technical world, but it is unclear how to represent and manage requirements on such ambient and deeply integrated systems. The group concluded that RE must become a bit more quantitative, dynamic, and adaptive in its analyses, in many cases get much more deeply into the modeling of domains which were traditionally outside the IT world. When attacking complex projects in the sense of John King, requirements engineers, developers, and also management stakeholders moreover must be prepared to get into deep research and very major, time-consuming and expensive conceptual and systems revisions as unexpected phenomena are bound to arise during such projects, unless they are very lucky. A stepwise approach to taking on and evaluating such systems in relatively simple steps is therefore strongly recommended, but unfortunately often prevented by the nature of the political debate.

4.5 Understanding Evolution: Biological, Cultural and Technological Perspectives

Lin Liu (Tsinghua University Beijing, CN)

Joint work of Liu, Lin; Myloupolous, John; Li, FengLin; Wang, Jun
Main reference L. Gabora, "The Origin and Evolution of Culture and Creativity,"Journal of Memetics – Evolutionary Models of Information Transmission, 1. 1997.

More and more software systems are required to cope with an ever- changing and evolving environment. These changes and evolutions could take place in the users needs and demands, in the capacity and availability of software assets, in the emergence of new technologies, and in the dimension of time and location, in the physical, systematical and social context of the operational environments. It has imposed great challenges for requirements engineering researchers to develop useful techniques to bridge the gap between intuitive user objective statements to concrete design constraints that is understandable and implementable by designers. In order to understand the evolution of systems and discuss requirements engineering strategies and techniques in response to the needs of evolution, we need to answer questions such as: what are the key challenges in requirements engineering for adaptation and evolution? What are the appropriate models for capturing changes? What are the levels of abstraction that should be used in light of adaptation and evolution? What are the possible types of adaptation and evolution? What are the appropriate strategies and mechanisms supporting requirements and systems co-evolution? I would set out from example evolution models in biological systems and cultural systems to give my observation on the general model of requirements and systems co-evolution.

5 Joint Panel with the Dagstuhl Seminar on "Foundations and Challenges of Change and Evolution in ontologies"

5.1 "When worlds collide: Requirements evolution and ontologies"

Matthias Jarke (RWTH Aachen and Fraunhofer FIT, DE) and Ulrike Sattler (University of Manchester, UK)

When we found out that in parallel to our seminar, another one with a very similar theme "Foundations and Challenges of Change and Evolution of Ontologies" would be organized by the knowledge representation/description logic community, we were excited to propose a joint session of both seminars to investigate how complexity, change, and evolution had been studied in these usually almost disjoint communities. We also hoped that some people from both seminars might identify joint topics they could collaborate on.

The joint session took the form of an open discussion with two introductory talks by Matthias Jarke and Ulrike Sattler. Matthias Jarke tried to establish the link from the RE side by pointing out the well-known formalization of RE by Jackson and Zave (1995) which was obviously inspired by the then very active AI community on model-based diagnosis : Given a set of domain assumptions D and a set of requirements R, find a suitable specification S such that $S, D \Rightarrow R$. In RE research, the R have often been interpreted as goals, to be refined and satisfied in some extended AND-OR graph structure. Matthias pointed out that reasoning in goal graphs is now well understood, and can in simple cases be reduced to SAT propositional reasoning. The fashionable research topic of provable compliance to regulations adds to the relevance of such approaches. As an even more direct bridge between RE and ontology, domain assumptions D are nowadays often described as more or less formal ontologies, at least in their static part. Thus, there seems to be a significant share of formalisms that seem relevant to both sides. However, since 1995, the problems in RE have shifted significantly, because the complexity and dynamics of systems has enormously increased. The dynamics especially in the business sector leads to frequent radical changes in the way domains are seen and modeled; witness the fashion industry as an extreme and obvious example, but also the new business models in the Internet, and the blurring of boundaries between hitherto separate industries e.g. in the Telekom sector. The RE seminar has identified three key challenges resulting from this growing complexity and change rate, which it would also like to pose to the ontology colleagues: At least the following key challenges to research and practice were identified in the seminar, together with counter-strategies where promising first steps for solutions:

- In large-scale projects, continuous changes in D and R lead to a shockingly high share of "black swan" projects that exceed budget by more than 70% and schedule more than 200%. A central cause are often politically motivated over-ambitious goals, with systematic under-estimation of requirements and resource needs, as well as poor change tracking. As a consequence, we strongly recommend not just to model the domain but also model and continuously monitor stakeholder goals, social structures and strategic dependencies during the development process, and even beyond. Monitoring and decision tracing should be supported by formal patterns (a kind of change ontologies?) to ensure compliance and effectively assess the impact of proposed changes.
- In the seminar, John King pointed out the difference between "complicated" and "complex" problems. Complicated problems can be solved by experienced, highly competent engineers with foreseeable effort. In contrast, complex problems can only be explored with uncertain results; thus, taking on a project that tries to solve a complex problem in one shot is bound

to lead to disaster. Platform strategies offer architectural mechanisms that constrain, but also leverage complexity. They constrain complexity because they offer a very large number of users and developers a uniform base understanding at a level of technology which is already well understood. They leverage complexity in that they allow end user-driven innovation on their top, witness e.g. the development of a huge app ecosystem on the iPhone platform. It could be interesting to see how ontologies might help to capture what the platforms offer, but one could also imagine data mining methods that would generalize an ontology of what is (successfully or not successfully) innovated by the huge chaos of apps.

After the social software revolution, few people would doubt that information systems need to be seen, and modeled, as socio-technical systems. But many traditional software systems are anything but social, and in the future, the upcoming cyberphysical systems reaching out to the real world with billions of sensors/actuators will teach us that our systems understanding so far is not sufficiently technical, either. The challenges mentioned above are therefore likely to increase in the near future, rather than being reduced. Perhaps a bit too immodestly, we claim that RE is the marketplace where responsibility is traded, and we need all the formalisms and tools we can get to help that task. Communication and mutual understanding is also the stated key goal of ontologies, so ontology evolution could of great interest of RE as well if a common language of interaction between the communities could be found.

Ulrike Sattler introduced the goals of the Ontology seminar. The knowledge representation/description logic community has focused on precise semantics for shared ontologies, supported by automated reasoning mechanisms with guaranteed time and space complexity, and a model theory that can operate purely at the schema level (possible worlds), or including instance-level real-world facts. One important specific objective has been to characterize precisely what kind of knowledge about the real world can or cannot be expressed by a specific ontology formalism. Research on change and evolution of ontologies has for a long time been a rather marginal part of the field, but is growing in importance in the last years. Four different formal approaches are being pursued:

- Calculating the difference between two ontologies (not just their syntactic structure, but also the derivable properties)
- Modeling timelines how concepts change over multiple versions of an ontology; as an example: In the fishery domain, the definition of what a salmon is has changed many times over the past 50 years; is it still possible to make, and formally justify a statistical statement whether the population of salmon has been growing or shrinking in this period?
- Belief revision within an ontology after new facts become known: numerous methods from databases (view maintenance, view updates) and knowledge representation (non-monotonic and paraconsistent reasoning) have been explored for cases where consistent repair of the overall beliefs is possible, or even where unrepairable inconsistencies remain.
- Reasoning about actions: Explicit modeling of actions and their effects aims at integrating the modeling of dynamic aspects into ontologies, thus supporting also tasks like planning and prediction from observed or modeled history.

In the subsequent discussion, many possible overlaps of interest were discussed. One example of shared interest is the law domain which has been a very interesting subject for ontology modelers for quite a while, and has become of great interest to requirements engineers due to the growing importance of all kinds of compliance in current systems. A second example is the question of dealing with paradigm change in fields such as biology, medicine, or even fashion, where the same data (biological specimen, medical observations, clothing colors and designs) need to be re-interpreted under new world views or ontologies – a problem also known in the field of databases as consistent query processing under schema evolution). A third interesting question is the comparison of virtual vs. materialized dependencies among domain and system concepts. Ontologies based on description logics that allow to represent also some aspects of incomplete or negative knowledge, might on the one hand help with automated reasoning about the relationships between requirements, or between requirements and designs, especially where the impact analysis of proposed requirements changes is concerned. On the other hand, traceability research in RE seems very advanced in materializing effectively the dependency knowledge where virtual reasoning would not be helpful because of human design decisions, or experience-based dependency structures for which no general logical description is available. One possible joint follow-up project which was envisioned in the seminar, could therefore be to find a new solution that combines the strengths of the KR and RE approaches in this field.

Participants

- Nicholas Berente
University of Georgia, US
- Richard Berntsson Svensson
Lund University, SE
- Jörg Dörr
Fraunhofer IESE – Kaiserslautern, DE
- Xavier Franch
UPC – Barcelona Tech – Barcelona, ES
- Gilbert Fridgen
Universität Augsburg, DE
- Anna Hannemann
RWTH Aachen, DE
- Sean Hansen
Rochester Institute of Technology, US
- Julia Heidemann
McKinsey & Company – München / Universiät Regensburg
- Jane Cleland-Huang
DePaul University – Chicago, US
- Jane Huffmann Hayes
University of Kentucky, US
- Matthias Jarke
RWTH Aachen, DE
- John Leslie King
University of Michigan – Ann Arbor, US
- Kim Lauenroth
adesso AG – Dortmund, DE
- Julio Cesar Leite
PUC-Rio de Janeiro, BR
- Lin Liu
Tsinghua University Beijing, CN
- Kalle Lyytinen
Case Western Reserve University – Cleveland, US
- Patrick Mäder
TU Ilmenau, DE
- John Mylopoulos
University of Toronto, CA
- Andreas Oberweis
KIT – Karlsruhe Institute of Technology, DE
- Barbara Paech
Universität Heidelberg, DE
- Xin Peng
Fudan University – Shanghai, CN
- Barbara Pernici
Politecnico di Milano, IT
- Balasubramaniam Ramesh
Georgia State University, US
- Isabelle Reymen
TU Eindhoven, NL
- William N. Robinson
Georgia State University, US
- Christoph Rosenkranz
Goethe-Universität Frankfurt am Main, DE
- Matti Rossi
Aalto University, FI
- Alistair G. Sutcliffe
Univ. of Manchester, GB
- Fan Yang-Turner
University of Leeds, GB
- Eric S. Yu
University of Toronto, CA
- Haiyan Zhao
Peking University, CN
- Andrea Zisman
City University – London, GB
- Didar Zowghi
Univ. of Technology – Sydney, AU