High-Impact Requirements for Software-Intensive Systems: A Manifesto

Matthias Jarke[1], Pericles Loucopoulos[2], Kalle Lyytinen[3], John Mylopoulos[4], William Robinson[5]

[1] RWTH Aachen University, Germany
[2] University of Loughborough, U.K.
[3] Case Western Reserve University, USA
[4] University of Toronto. Canada
[5] Georgia State University, USA

Abstract. Despite its undoubted success in the last two decades, requirements engineering needs a better alignment between its research focus and its grounding in practical needs as these needs have changed significantly. We identify and explore changes in the environment, targets, and the process of requirements engineering (RE) that influence the nature of fundamental RE questions. Based on these explorations we propose four key principles that underlie current requirements processes and influence their successful resolution: (1) intertwining of requirements with implementation and organizational contexts, (2) dynamic evolution of requirements, (3) architectures as a critical stabilizing force, and (4) high levels of design complexity and necessity to employ new ways to mitigate it. We make recommendations to refocus the RE research agenda as to meet better emerging and new challenges based on the review and analysis of these four key themes, and note several managerial and practical implications**.**

1. Introduction

Requirements engineering research gathered momentum around 30 years ago. Its genesis was motivated by practitioners who noticed the urgent need for systematic requirements engineering in large software projects of the 1970's when they became increasingly troubled[1, 2]. Much of RE research since then has focused on artifacts that help capture, share, represent, analyze, negotiate, and prioritize requirements as a basis for design decisions and interventions. Its main outcomes and findings are evidenced by the volume and impact of a plethora of papers published in the Requirements Engineering Journal and in the IEEE Requirements Engineering conference series, as well as in leading software engineering journals including IEEE Software, IEEE Transactions on Software Engineering, ACM Transactions on Software Engineering and Methodology, and Communications of the ACM. Due to its practical origins it is not surprising that some of its findings, like the use of business and system modeling (ERD, use cases), risk driven methodologies, structured requirements documents, and simple requirements tracing, have found their way into practice [3].

Yet, the environment in which software engineering and requirements engineering is practiced has changed dramatically over the past 10-15 years. Partly, this is due to increases in computing speed, lowering of computing cost, and advances in functionality, which has made software. Partly, this is due to the changes in technological, task, and organizational environments where software is either produced or deployed. The field's focus and scope has, as a result, shifted from engineering of individual systems and components towards the generation and adaptation of IT-supported ecosystems. Accordingly, a term *design requirements* rather than *software requirements* is needed as an inclusive term to denote all common sets of requirements issues within these ecosystems that need to be addressed at the crossroads of business, software engineering, and industrial design. This shift has created a strong need to re-think and re-align established RE practices to meet the new practical challenges. To this end both academia and industry need to understand better critical issues that underlie current requirements engineering and address associated new challenges. We posit that answers cannot come just from doing more of the same—i.e., traditional RE research focusing on notations and tools alone. The research scope of RE has to become more interdisciplinary and it needs to also carefully evaluate some of its critical assumptions that underlie current research and practice.

This essay aims to identify some of these challenges based on a detailed field [4] and content analysis of extensive expert discussions and feedback on current requirements engineering practices[1]. Based on these explorations we propose four principles that underlie future requirements processes and influence their successful resolution. Finally, new research challenges and practical implications are identified.

2. The Changing Nature of Requirements

This essay is motivated by the perception that current requirements engineering is marked by a range of new challenges and opportunities[3]. Its environment, target technologies, processes, and fundamental problems have undergone a tectonic shift.

The environment of RE now involves elements that were not there 20 or 30 years ago[5]. First, the economics of RE has changed. Large systems like ERP systems need more rigorous ROI, but at the same time horizons for ROI have reduced to 18-20 months made feasible by massive reuse and COTS deployment. Second, there is practically no green-field software development; RE acts more like the ancient Roman god of gates—Janus, with one face, looking at new business and technological challenges and opportunities and another face gazing at existing (technological, organizational, social and political) environments. Third, the scaling towards IT supported ecosystems results in exceedingly complex and non-linear dynamic dependencies between system components and their natural, technical and social environment—

[1] These were recorded in a June 2007 workshop held in Cleveland, U.S. and an October 2008 perspective seminar at Dagstuhl Castle, Germany. See appendix 1 for a more detailed description of the study design and data analysis.

“green IT” being just one of the latest buzzword that characterize this trend. Fourth, speed and agility, time to market, low-cost iterative or even end-user development have become critical factors leading to the need to search for new trade-offs between efficiency, openness, and flexibility during requirements capture. This has also increased outsourcing and off-shoring, which requires disciplined evolution and management of explicit specifications as a basis for delegation. Fifth, RE now cuts across industrial design (e.g., pervasive applications), media design (e.g., e-commerce and media applications), interaction design (e.g., new modalities of interaction), and business environment design (e.g., open business platforms). Overall, design requirements need to capture and coordinate such increasingly diverging and dynamic needs of users and other stakeholders during the life-cycle of a product, a service or a platform.

What are the critical issues that emerge during RE in this brave new world? Table 1 presents nine critical issues that were solicited in a field study involving 39 top level RE professionals in Fortune 500 companies[4]. These are divided into the changing nature of the object of RE (target platform), and the process of RE (development process).

Table 1 Summary of Critical Design Requirements Issues (expanded from [4])

	Critical Requirements Issues	Brief description
Target platform	**Business process focus**	Requirements focus simultaneously on the business process, and requirements for technological artifact driven by that business process.
	Systems transparency	Requirements involve the demand for a seamless user experience across applications.
	Integration focus	Requirements focus on integrating applications rather than development of new ones (i.e., less green-field development).
	Packaged software	Purchase of commercial off-the-shelf (COTS) software and components rather than internal development. This has lead to market driven vendor-led requirements and knowledge brokering.
Development process	**Distributed requirements**	In addition to diverse stakeholders, requirements processes are distributed across organizations, geographically, and globally.
	Centrality of architecture	Architectural requirements take a central role and drive business, product and application requirements.
	Layers of requirements	Requirements are iteratively developed across multiple levels of abstraction, design focus, or temporal horizon.
	Interdependent Complexity	While some forms of design complexity have been reduced, the overall interactive complexity of the design ecology has risen significantly.
	Fluidity of design	Requirements process must accommodate the need for continued evolution of the artifact and the solution after initial implementation.

Overall, these issues resonate well with the debate around Simon’s design classic, *The Sciences of the Artificial*[6]. On the one hand, they show that software designs resemble increasingly continuous and dynamic searches for satisficing solutions—not an optimized and fixed solution at one time point conforming to a fixed set of requirements. On the other hand, they go beyond Simon's model in that they emphasize sensemaking and problem framing in complex uncertain environments [7] over problem solving in a bounded context. To wit, changes in the environment and the object and process of RE have changed the three key RE problems as follows:

First, the fundamental *design requirements problem* can now be stated as follows: *What is the emergent behavior and dynamics of the software artifact and its environment in their evolutionary trajectory?* Accordingly, designers and other stakeholders need to ask: will the system continue to satisfy emergent goals, and what those goals could be expected to be during the artifact's life-time?

Second, the *specification problem* can now be stated as follows: *How can designers anticipate and represent the emergent behaviors of the system and its components and how does the resulting system behavior conform and relate to the emerging environments and the notations used to represent and predict it*?

Third, the *predictability problem of designs* can now be stated as follows: *how does the artifact and its behavior change the environment as to make our predictions of the system behaviors faithful*? In other words, how does the dynamic composition of the system and environment together differ from the environment alone, and can we predict well the impact of the system on the environment, and vice versa?

3. Four Requirements Principles

The revised key problems of RE invite us to re-think key principles that underlie RE and design practices. The RE research has been informed over the last two decades by a few such principles as the idea of information hiding, or the principle of abstraction. These principles helped reduce design complexity, localize design decisions and reduce their interference, and analyze and predict system behaviors and structure during design from specific viewpoints. But what are the key principles that will underlie successful design and RE in the new world we face? What principles will help us address the design requirements problem, the specification problem, and the predictability problem? We propose next four principles that were gleaned from our analysis of expert opinions and a review of the related literature. These four principles are:

1. **Intertwine Requirements and Contexts:** The necessity to *intertwine* design and requirements with design and implementation across multiple dimensions.
2. **Evolve Designs and Ecologies:** The necessity to view design and design processes as *evolving* elements in *an ecology.*
3. **Manage through architectures:** *Architectures* have a critical role of as enablers and constraints in the constant creation and shaping of design ecologies.
4. **Recognize complexity:** The heightened *complexity* of requirements processes demands new ways to approach design problems and manage requirements.

We will next describe each principle in more detail in terms of the nature and content of the principle, related research questions and emerging research to address these research questions.

3.1. Intertwine Requirements and Contexts

The debate about the role of requirements in a system's development process is as old as the field itself. Whilst a rough consensus has been reached, that requirements are a pre-requisite for downstream development, there is a great deal of inconsistency and controversy on how 'problem' and 'solution' spaces interplay during the evolution. One school of thought regards the influence of implementation on requirements as being harmful[8]. They argue that understanding the system's context, such as its organizational and social factors and goals can provide a sufficient set of functional and non-functional requirements, which can then be mapped onto appropriate implementation models and platforms. This perspective regards requirements as the bridge between the 'subject' and 'system' worlds assuming that there exists a high degree of stability on business, organizational, and community goals. An opposing view stresses the need for revisiting requirements as implementation progresses and emphasizes the dynamics and intertwining of these activities [9].

The review of existing practice [4] shows that implementation and its requirements specification are now necessarily intertwined as many requirements emerge from existing solution spaces. Accordingly, the concept needs to be extended to the whole system context. In many situations, the salient factors shaping RE seem to be innovation and effective differentiation and interplay between the two worlds has become more intricate, complex, dynamic, and generative. The traditional idea of an enterprise that creates wealth by getting better at improving its existing products and services through for example business process re-engineering, improved supply chain, better cost control is now constantly challenged by approaches which combine research, creativity, and human-centered management practices. In these innovation-driven settings, requirements become part of both the business solution and the system solution. They become critical in constantly bridging new solutions to organizational and societal problems. The evolving designs need to reduce the distance between a problem and a solution through novel and dynamic thinking, acting, and innovating. In such a design-thinking culture, design requirements become increasingly central and need to be understood as part of a multi-system, socio-technical ecology, which drives innovation. Software requirements need to be dynamically and constantly situated between these spaces as they intertwine organizational and implementation considerations, providing leverage to influence both.

It also may be the case that some systems may be reaching a practical *world-model limit* due to the constant intertwining. While prior design efforts could rely on an adequate, stable, world-model as the basis for specifying nearly stable software designs, now, software must be agile—rapidly evolving to meet new and changing needs. The scope of stable world-assumptions is more limited for context-aware, customer-focused applications. The idea of evolutionary software and variability selection aims to meet these needs. But little attention has been given to the challenge of developing evolutionary world models that form the basis for the necessarily simplifying model assumptions in the software. The fundamental intertwining between requirements and contexts constantly seeks some correspondence between the models in software and its world context, as software that is based on an adequate, flexible, and evolvable world-model is more likely to survive the challenges of intertwining, evolution, and complexity. Dealing with the world means that software and its developers must monitor and evolve their understanding of the world as an adequate correspondence between the world and the modeled world needs to be maintained. Therefore, requirements processes face a new kind of uncertainty that goes beyond traditional RE uncertainty, which has been characterized by: (1) requirements identity (knowing requirements), (2) requirements volatility, and (3) requirements complexity[10]. To mitigate this new kind of RE uncertainty, developers need to devote their efforts to address: (1) *requirements fidelity uncertainty*, which denotes the uncertainty about the level of intertwining between the world and the software model. Examples of techniques that help mitigate fidelity uncertainty are exception and event-based analysis; software tailoring and user-based development, and case-based learning; and (2) *requirement monitoring uncertainty*, which denotes uncertainty of the level and mode of observation and analysis necessary to assess the world, the model, the requirements, and their alignment. Examples of monitoring include ethnographic methods, business activity monitoring (BAM), and software instrumentation with automated monitoring. These two new levels of uncertainty highlight the need for run-time monitoring of the software context to maintain the fidelity of the world-model intertwining with requirements through time. Though all these steps are promising, we need more research on how to analyze and understand the design complexity and how to deal with it.

3.2. Evolve Designs and Ecologies

Meeting stakeholder needs is a fundamental to requirements activity. Given that requirements are increasingly intertwined with organizational and implementation concerns, they must constantly evolve to meet changing needs and emerging implementation alternatives. Traditionally, causes of evolution have been classified into four classes: (1) the software, (2) the documentation, (3) the properties of the software, and (4) customer-experienced functionality[11]. Evolution has been accordingly studied as a software design problem addressing its methodological support (e.g., how can development activities most effectively incorporate evolution?) and management activities (e.g., how one can one record and trace software releases or link the code to changing domain knowledge?).

Software development involving globally distributed software teams across multiple sites raises also new issues of communication, coordination, and control associated with such evolution. Activities in open source development, such as inter-project merging and the creation of new software artifacts, compound the need for new methodological frameworks to cope with requirements evolution. The reality of an ever incomplete and evolving design needs addressing. What are the principles that can guide developers to generate and accept ‘incomplete designs’ as being functionally sufficient whilst maintaining its flexibility to be extendible?

For example agile methods and scenario based modeling offer a means to better cope with the fast paced creation and evolution of requirements ecologies[12]. Likewise research into co-evolution and co-design[13] has sought to address drivers and interaction laws that deal with the intertwining of contexts and requirements. Yes, such studies are clearly in early stages and agile methods only deal with micro-level evolution of local system development tasks but ignore recursive nature of change as it propagates across higher level architecture and system layers. Here we clearly need more longitudinal studies of the dynamics and change in software ecologies and how different causes ranging from technological, user level learning, organizational policies, market based, and regulatory changes intertwined and generated specific evolutionary paths in the system evolution. What are appropriate co-evolutionary design methods? How does one determine the impact of co-evolutionary design change?

3.3. Manage through Architectures

Architecture is concerned with design blueprints that connect high level organizational, business, or implementation considerations with a long-term evolutionary perspective. Organizations now increasingly conform to business or information architectures that provide stability, scale and change to their business rules, data and models. Designers have for some time relied on implementation architectures while evolving their designs. In its variety of forms, architectures provide the stable stepping-stone necessary to understand and evolve any system across different domains. Through release planning, requirements play a central role in systems evolution, while architectures provide "nearly" fixed points of reference to moderate, constrain and enable evolution. Such dependence on architecture may be inferred from Lehman's law that "the incremental growth (growth rate trend) of evolutionary software systems is constrained by the need to maintain familiarity" [14]. Architectural dependences arises in the approaches of IKIWISI (I'll Know It When I See It) and COTS (Commercial off-the-shelf) software deployment [15].

Though RE research and software design in the past has paid significant attention to 'software architectures' they offer limited insight to the role and use of architectures in the new RE practices. Many of these studies focus on identifying and organizing specific design elements and their components in the context of a single system. Accordingly, they approach architectural design similar to generating a blueprint of a single house (or even part of it). In the context of dynamic and evolving ecologies, such an analogue fails. In the new world, management through architectures is about generating multiple and multifaceted plans similar to urban planning. Like in urban planning, the architectures in RE may have alternative and overlaying *variation points* that influence the evolution of the whole software ecology; they also integrate to varying degrees the needs and interests of different stakeholder groups with different roles; and finally they involve the same level of complexity and interdependencies. Like urban planning blueprints, architectural models also provide the key artifact for coordinating applications, functionalities, and their evolution. Finally, like urban plans they embody specific business models or design visions, and can come in different forms in different design contexts. In the new RE, we see an increasing variation between types of architectural models needed, i.e. how they relate to specific families of systems.

Recent attempts to deal with architectural considerations include the transfer of industrial concepts, such as product lines or platforms to the software field [16]. It is apparent that we need architectures in various forms to manage and cope with continuous and rapid change by defining the scale, scope, and direction of variance and selection. They help stakeholders to envision the impact of proposed changes by providing contextual information and allow for fixing and selecting variation points across multiple stakeholders. Such contextual information across multiple domains is typically far more stable than other aspects of a designed system and its ecology. Yet, many research challenges prevail in taking advantage of the idea of architectures: How do architectures influence the evolution of requirements and their identification? What is the nature of requirements discovery and elicitation under different architectural principles and information? Is it possible or desirable to construct a common ontology of business, information and technology architectures, and how to relate them? How can architectures be exploited to aid flexible composition of systems and ecologies?

3.4. Recognize and mitigate against design complexity

Complexity is borne out of the existence of multiple uncertain futures that relate to software and their evolving ecologies[17]. A mix of human, social, political, economical, technological, and organizational factors has a bearing on the level of complexity associated with requirements tasks. Dealing with design complexity in the ecologies impacts two areas: (1) the strategic decision-making in generating and selecting requirements and understanding their impact on the future system's ecology, and (2) selecting tractable design-approaches that make complex system designs possible. The former is the concern of how to relate complexity with stakeholders within their 'subject' world, whereas the latter influences behaviors within the 'design' world.

Overall a new sort of environmental or interaction complexity[2] needs to be reckoned and managed during the RE process. The implementation of requirements impacts not only on the technical systems, but also on their organizational and social settings and increases their interactive complexity. In addition, the increased variety of requirements that emanate from diverse communities need to be negotiated, evaluated, and selected. Qualitative and often structured conceptual models like goal or business models, whilst rich and useful in representation and analysis for design are less helpful for stakeholder evaluation and understanding the interactive complexity. Due to the design complexity, it is also difficult for some stakeholders to visualize and understand the system's behavior from observations or during walkthroughs. It is tempting to think that "… stakeholders understand a description when they don't really understand it at all". Many research questions remain poorly understood concerning design complexity: What is the nature of design complexity and how can we analyze and measure it? What types of interdependencies influence and affect system change and create higher levels of design complexity? How can architectural models be exploited mitigate against design complexity, and to what extent they are a cause of it?

One way of coping with new levels of design complexity is through architectural designs and control that allow 'nearly' decomposable system designs. This mitigates complexity by ensuring that interactions among components are weak, though not

[2] This should not be confused with computational complexity as defined by well-known complexity notions like NP hard problems.

negligible. Design is perceived as a problem solving and framing process with inherent uncertainties driven by the partially unknown unknowns in the context and implementation. Thus, designing a nearly decomposable system in the face of uncertain requirements becomes a difficult satisficing problem. Perhaps, not surprisingly, a new look at design methodologies can play the central role here. An emergent design methodology will be an improvement over conventional *a priori* methodologies. Open source systems, following nontraditional methodologies, for example evolve systems faster than traditional life cycle and requirements driven development approaches [18]. The co-design and co-evolution of the system and its stakeholders seems to play here a central role, as does fact that "open source systems entail internal architectures with orthogonal features, sub-systems, or modules, as well as external system release architectures that span multiple deployment platforms" [18]. A better understanding of complexity may also be obtained, if system descriptions are tested against concepts with which stakeholders are familiar and multiple scenarios are played out by fixing key parameters as proposed by the architecture. Experimenting with different scenarios as a way of postulating possible design alternatives has proved a powerful means for discovering and refining requirements with heightened complexity [19].

3.5. Summary of Four Key Design principles

The four principles are summarized in Table 2 together with the rationale for using each principle as well as what critical RE issues motivate each design principle.

Table 2 Four key requirements principles.

Principle	Description	Rationale
Intertwine Requirements and Contexts	Requirements are interdependent with their social and technical contexts. As boundary objects in the intersection of the technical and social domains, design requirements seek to constantly resolve the gap between problems and solutions. Specification and implementation intertwining is long recognized, but the social context and specification intertwining is growing in importance. *RE Problem addressed: design requirements problem.*	Intertwining between business, organizational, community context and requirements is as important as it is between requirements and software. *New RE Issues : Fluidity of designs, , Business process focus; Integration focus, Distribution of requirements*
Evolve Designs and Ecologies	Design ideas and artifacts evolve, from stakeholder preferences to the implementations. Evolution needs to be managed through selectively freezing some aspects while changing other aspects thus allowing increased variation, dynamic selection and diffusion of structures and behaviors. *RE Problem addressed: design requirements problem, specification problem.*	Everything evolves, but at different rates. Therefore design around relatively fixed evolutionary paths that allow for increased but controlled variation and effective selection and diffusion. *New RE issues : Fluidity of designs, Layers of requirements, Distribution of requirements, Packaged software*
Manage through architectures	Architecture is the least evolving and most widely referenced anchor aspects of the design, be it business architecture or implementation architecture. *RE Problem addressed: specification problem*	If well-designed, the architecture (business or software) evolves slowly, and influences and interacts with many requirements. We know poorly however how architectures shape, allow and constrain evolution. *New RE issues : Interdependent complexity, business process focus, Centrality of architecture*
Recognize and mitigate against design complexity	The necessity to simultaneously consider a large number of issues and their non-linear interactions during design raises design and requirements complexity beyond what a single designer can understand or visualize. *RE Problem addressed: predictability problem.*	Historically, tools aided a single designer or a small group in decision making while tracking and analyzing the current state of design. New design tools need to be extended to monitor and analyze the dynamic design evolution, highlighting its trajectory and helping negotiate at the team and community level. *New RE issues : Interdependent Complexity, Fluidity of designs, Business process focus, Layers of requirements*

4. Implications

In this manifesto, we argue that current and future design requirements are shaped by the rapid change in implementation capabilities and platforms, new application demands, and rapidly evolving environments. Over its thirty-year history, the idea of design requirements has changed from single, static and fixed-point statements of desirable system properties into dynamic and evolving rationales that mediate change between the dynamic business environments and the design and implementation worlds. As Fred Brooks noted in the Dagstuhl workshop: “Design is not about solving fixed problems; it is constant framing of solution spaces”. This evolution has now probably reached a new turning point characterized by unprecedented scale, complexity, and dynamism. This calls for new ways to think about requirements and their role in the design. Like earlier turning points, such as the software crisis in the 1970s, it will demand a resolute and careful intellectual response. The four requirements principles discussed in the last section have numerous implications for research questions of which only a minority has been addressed yet. The Appendix lists a number of research topics and individual research issues collected in the workshops we conducted.

There are also implications for requirements engineering practice both at the management and at the engineering level. For the sake of brevity, we discuss three closely related strategies to deal with the issues exposed by the four principles.

Service orientation and outsourcing: The information systems life cycle must be aligned even more closely with the business process lifecycle. Service oriented architectures, possibly combined with model-driven code and test generation are beginning to be reasonably well established at the programming level. However, the situation is still quite different at the level of business services, despite ongoing standardization efforts. The decomposition of monolithic business process management systems into freely configurable business services has turned out to be far more complex than expected due to the need to make business semantics explicit that was hitherto hidden in the code. This is, however, not only true in runtime service configuration, but also when outsourcing and especially off-shoring of development tasks goes beyond simple mass customization and enters the domain of business semantics, which is often also culture dependent. Intercultural competencies become a must for requirements engineers in such settings, with pilot case studies being a promising means to establish that understanding. Legal aspects are also an increasingly important aspect of requirements engineering, not only in terms of protecting intellectual property (IP)—what should I *not* offshore if IP handling is doubtful—but also in terms of protecting against being sued by customers due to imprecise contractual agreements.

Importance of the Edge: In the increasingly complex and rapidly evolving environments the classical distinction between users and developers is vanishing and both the user network and the developer network are becoming increasingly fuzzy at the boundaries. Many contributors to system designs and even implementations are no longer located in the kernel teams, but at the edge of these networks. This situation—which has been characterized as an evolution from *user* to *citizen*—also enables much greater diversity of system variants and system uses, especially in cross-cultural environments. Web-based social networks have proven to be the infrastructure of choice for such settings, and quick and dirty beta testing of incremental changes in limited market portions is the requirements testing strategy of choice.

The incentives typically go beyond purely monetary ones. Industrial design aspects such as innovativeness and aesthetics in the user experience often play a larger role than pure functionality, as well as ethical and legal considerations. Bricoleurs at the edge must be harnessed to contribute in terms of usability and technology, but they must also be kept motivated to do so beyond the initial phase of enthusiasm, and management must be able to deal with situations when for whatever reason they leave the network. Transparency, accountability, and maintenance of a core vision in such systems become even more important than in traditional applications, so requirements traceability within the design and evolution process, but also runtime monitoring at the requirements level are becoming critical issues for management and RE practice.

Capability-based organizations and platforms: Organizations react to these challenges by increasing variability in the space of their actions. For the past twenty years, the dominant paradigm has been business process modeling and optimization together with related often monolithic standard software packages, or more recently software product lines. However, we are now witnessing a move towards emphasizing capability-based organization networks over process organizations. Such organizations or networks analyze or define their core capabilities, and seek opportunities (often initiated from the edge, as discussed above) to exploit them for market or process innovations in widely varying scenarios.

In terms of IT strategy, capability-based organizations are closely related to platform concepts through which the core capabilities can be competitively delivered and the network held together while allowing for great flexibility in the details. By fixing certain requirements in efficient core implementation, the platform of course bets on certain assumptions about the speed of change in different kinds of requirements. For example, platform strategies have for a decade been a critical success factor of large companies in the automotive industries but this advantage turns into a deadly trap to organizational adaptation when economic considerations mandate that the company be split in a manner orthogonal to the definition of the platforms. Software platform strategies as a means to manage efficiency and variability beyond the conceptual level offered by software product lines need to consider a distinction between core processes (in the platform) and context processes (around it), they must be based on a careful analysis of market power and partner network strength with respect to a given abstraction of the market situation, they must cover foreseeable technology evolution also in the underlying technological standards.

Overall, the good news is that the importance of RE continues to grow as the arguments for it are broadening. But the bad news is that besides the need for making a business case for a decent return on investment within a shorter time frame, in the future RE we need to consider additional arguments such as the need for the alignment with business process, understanding your own capabilities, systematizing the customer expectation managements, ensuring legal protection against IP loss or contract violation suits, creating user buy-in, and minimized training costs when justifying your next RE project. We need to therefore expand RE research into new fields—including complexity science, industrial design, organization design, and economics, among others. One challenge is the void of interdisciplinary intellectual exchange between diverse communities that have a stake at software requirements given the observed need for increased diversity in the design of software-intensive systems.

Acknowledgments

We thank Sean Hansen, Nicholas Berente, Dominik Schmitz, and Anna Glukhova for helping to organize the workshops, the workshop participants for inspiring discussions, and Sol Greenspan for constructive comments. This research was in part funded by National Science Foundation's "Science of Design" initiative, Grant Number: CCF0613606, by DFG project CONTICI and by BMBF project ZAMOMO (01 IS E04).

References

1. D.T. Ross and K.E. Schoman Jr., "Structured analysis for requirements definition", Transactions on Software Engineering, vol. SE-3, no. 1, 1977, pp. 6-15.
2. J. Frederick P. Brooks, The Mythical Man-Month, Addison Wesley, 1995.
3. K. Lyytinen, et al., Design Requirements Engineering: A Ten-Year Perspective, Design Requirements Workshop, Cleveland, OH, USA, June 3-6, 2007, Revised and Invited Papers, Springer-Verlag, 2009, p. 495.
4. S. Hansen, et al., "Principles of Requirements Processes at the Dawn of 21st Century," Ingénierie des Systèmes d'Information, vol. 13, no. 1, 2008, pp. 9-35.
5. D. Schuler and A. Namioka, "Participatory design," Proc. Lawrence Erlbaum Assoc., 1993.
6. H. Simon, The Sciences of the Artificial, MIT Press., 1996.
7. D. Schon, The reflective practitioner: How professionals think in action, Basic Books, 1983.
8. J.P. Bowen and M.G. Hinchey, "Ten Commandments of Formal Methods," 1995.
9. W. Swartout and R. Balzer, "On the inevitable intertwining of specification and implementation," CACM, vol. 25, no. 7, 1982, pp. 438-440.
10. L. Mathiassen, et al., "A Contigency Model for Requirements Development," Journal of the Association for Information Systems, vol. 8, no. 11, 2007, pp. 569-597.
11. N. Chapin, et al., "Types of software evolution and software maintenance," Journal of Software Maintenance and Evolution Research and Practice, vol. 13, no. 1, 2001, pp. 3-30.
12. A. Cockburn, Agile Software Development, Addison-Wesley, 2002.
13. C. Berger, et al., "Customers as co-designers," Manufacturing Engineer, vol. 82, no. 4, 2003, pp. 42-45.
14. M.M. Lehman, "Software Evolution," Encyclopedia of Software Engineering, vol. 2, 2002, pp. 1507-1513.
15. B. Nuseibeh, "Weaving together requirements and architectures," Computer, vol. 34, no. 3, 2001, pp. 115-119.
16. K. Pohl, et al., Software Product Line Engineering : Foundations, Principles and Techniques Springer, 2005.
17. M. Godet, Scenarios and strategic management, Butterworth-Heinemann, 1987.
18. W. Scacchi, "Understanding Open Source Software Evolution," Software Evolution and Feedback, Theory and Practice. Wiley, NY, 2006.
19. J.M. Carroll, "Scenarios and design cognition," Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on, 2002, pp. 3-5.
20. B. Glasser and A. Strauss, "The development of grounded theory," Chicago, IL: Alden, 1967.
21. M. Jarke, K.Lyytinen, J. Mylopoulos (eds.): Science of Design -- High-Impact Requirements for Software-Intensive Systems. Dagstuhl Seminar Proceedings 08421, http://drops.dagstuhl.de/portals/index.php?semnr=08412.

5. Appendix: Data collection and analysis

The new requirements engineering challenges reported in this manifesto were identified, discussed and reviewed in two workshops, designed to accomplish three objectives: (1) engage separate research communities in a dialogue, (2) strengthen design science principles, and (3) open new vistas for the research on design requirements. The first workshop was held in Cleveland, United States, in June 2007 (http://weatherhead.case.edu/requirements), while the second workshop was held as Perspective Seminar in October 2008 in Dagstuhl, Germany [21]. In addition we undertook a field study across a range of industrial and systems design settings to understand the perspectives of practitioners on successful requirements practices, anticipated developments, and new challenges in design environments [5]. In the study we sought participation from senior technology leaders within a range of Fortune 500 organizations and leading software developers. Semi-structured interviews were conducted with 39 individuals on new challenges related to RE practices and design. The sampling approach reflected purposeful bias toward large, complex systems in an effort to focus on practices associated with the most challenging development contexts. The systems development efforts reflected in the study involved from tens to hundreds of man-years. System costs ranged from several million to hundreds of millions of dollars. All interviews were transcribed to support formal analysis of the interview data.

The focal contexts of the studied systems and their requirements included the following:

- *Large, complex organizational information systems*: The design of very large information systems, often supporting inter-organizational exchange of information; including transportation systems, distribution networks, and defense contracting.
- *Embedded systems*: The design of systems and components intended for integration within broader design artifacts; includes automotive and aerospace design environments.
- *eBusiness Applications*: The design of artifacts and information systems for use within a Web-based delivery channel; includes portals, e-commerce establishments, and other Internet-oriented product and service providers.
- *Middleware Systems*: The design of integrated software platforms that support the exchange of data between distinct applications.

The two workshops in Cleveland and Dagstuhl involved presentations of short papers that focused on one or more facets at the cutting edge of requirements theory and practice. Elaborated versions of these papers can be found in [3, 21]. The following issues were selected for deeper exploration in working groups and plenaries:

1. Strengths of previous requirements and design research in different disciplines
2. Analyses of contemporary practices in the form of case studies, field studies, and experience reports of successful or failed practices
3. Identification of emerging critical issues in the design of complex software-intensive systems
4. Opportunities for intellectual cross-fertilization across disciplines in design and requirements

5. Identification of new avenues in requirements research by mapping out a new landscape and emergent challenges
6. Identification of new theories and research methodologies pertinent to the emerging challenges

Building upon these initial themes, workshop participants were asked to reflect on their research and practical experiences, and identify and characterize areas that would deserve greater attention. Participants presented and explored themes refined from Table 1, some key findings and observations in the form of research questions are shown in Table 3 based on a content analysis [20].

Besides elaborating these themes, the Dagstuhl Workshop discussed also some of the most important identified challenges in more depth:

7. Interlinking of multiple concepts of design, and its intertwining with requirements
8. Evolution and management of requirements under growing complexity
9. Architectural implications and platform strategies
10. Identification of changed stakeholder roles and management arguments

Additionally, the Dagstuhl Workshop identified specific implications for research and practice of developments, such as service orientation, growing importance of the edge, and new capability-based organization forms were identified and explored.

Table 3 New Design Requirements Topics.

Topic	Topic description	Research questions
1.New Concepts of Design	**Fundamentally new design concepts and processes that enable design are emerging.** **The role and function of requirements in generating and assessing design are changing.** **Challenges and opportunities associated with diverse design theories and their associated requirements will drive design research.**	• How to put multiple owners of problems "in charge" of their problems? • How to explain or analyze adequately ill-defined problems, and how does externalizing some of them improve the design ability? • How to design an ecology instead of a bounded system? • If we say requirements involve more designed system, how can we still remain relevant for design? • If requirements encompass the idea of an open, evolving system how can we design with an idea of a fixed and closed set of requirements? What do open requirements mean and how do we manage them effectively?
2. Fluidity and incompleteness of Design	**Requirements evolve constantly from the perspective of design participants and other stakeholders** **Evolution implies fluidity of designs resulting in incompleteness in design requirements and highly adaptive systems.** **New interactions between models and systems as enabled by run-time evolution** **The transience of requirements within design affects designer and stakeholder learning.** **Drivers for requirements change are becoming more diverse and critical. This affects requirements negotiation and designers need to recognize critical killer / non-negotiable requirements**	• How do environments evolve designs as the environments constantly and increasingly change? • What changes requirements are due to the introduction of new technologies through socio-technical design? • What is maximum scope and rate of change that is sustainable? • What types of new "informalisms" we need to make sense of the change, in addition to old formalisms? • What types of run time mechanisms one needs to monitor and evaluate achievement of the requirements? • How can we formulate and choose good design metrics; and create a "design dashboards for fluid designs"? • Requirements come often into being after implementation. How can we make sense of emerging webs of discourse that negotiate and make sense what are the directions of change? • What are the consistent properties (constraints) i.e. the "design anchors" that help keep design stable? What can become the basis of sustainable incomplete designs? • How to integrate fixed and soft requirements and best effort designs based on evolving preferences? • How to adapt to new requirements or satisfy requirements through continual software based monitoring and evaluation • Can organization design anymore be separated from infrastructure and information system design?

Topic	Topic description	Research questions
3. Visualization, Representation and Analysis of requirements	**Visualization and information representation play a critical role requirements processes. They shape requirements discovery, organization, validation and verification.** **New requirements features necessitate to apply powerful capabilities that allow visualization of requirements and systems across stakeholders and design contexts**	• What models and visualization schemes are needed to represent and analyze complex model interactions or to locate and discover requirements? • Design artifacts and their meanings are social. What new representations we need to offer a means to negotiate meanings and sense-making across different groups? • Today we have one person and many computers, and we deal with distributed computing environments, which serve multiple people. How can we represent these communities and their growth in our theorizing and analysis?.
4. Managing Complexity	**Systems increase in their complexity and designs involve increasingly management of interdependencies between elements. The complexity is addressed by improved modularity, but this increases system interactions and scale and thus increases complexity.** **Design ambiguity and uncertainty has increased and they affect requirements discovery.**	• How can we address questions of centralized control v.s. distributed information requirements in different and new ways as the complexity increases? • Requirements emerge often due to increased specialization. This allows on one hand for removal of complexity and its black-boxing. But, at the same time this requires understanding principles of modularity. How can we achieve balance between modularity and specialization when the complexity and dynamism increases?
5. Business model /Process Focus	**Business process or related experience is the key unit of requirements analysis.** **Process design is becoming the cornerstone of requirements discovery and a means to bound or enable design efforts.** **Ways of making sense and representing business models affect process designs.** **Requirements are increasingly grounded on higher level business models and business process architectures including industry wide generic process and data standards.**	• How can we detect and deal with new type of indeterminism when one must deal with a multitude of business rules? • All the issues one traditionally embedded into operating systems, one now gets on the organization level (scheduling, etc.). How do you handle that? • The power of process modeling may not be in making a single and unified model, but in generating multiple models across an organization as to cross check them. How do you address the use and semantics of such models? • How to look for the "meta-patterns" for business design and cull that from the user dialog? • Business models are scaling up; there is no distinction between enterprise and business process level models. How can you address the higher level of granularity and change in such models? How do you validate them? • How does one model, analyze, and align new business models and associated software models like e.g. software platforms?

Topic	Topic description	Research questions
6. Stakeholder Issues	**Stakeholder's role in requirements efforts is changing and diversifying.** **Roles including end user based learning and need discovery.** **New forms of engagement are emerging including: wide area user participation; community driven requirements discovery; new participation roles and duties; changing governance, new incentives, and varied forms of negotiation and conflict resolution.**	• How can requirements processes and stakeholder involvement embrace both technical and political considerations at the same time and address simultaneously technical and political requirements • How do we navigate through design spaces when different stakeholders have a different understanding of terms and world assumptions? • Can different design artifacts be used to make some stakeholder believe that they have more shared understanding than actually prevails?
7. Impact of New Technologies / Architectures	**New technologies and architectures shape design spaces and influence the identification, documentation, and management of requirements;** **New interactions between legacy infrastructures and emerging technologies including the discovery and projection of new requirements from technological opportunities,**	• Architectural plans in enterprise wide systems in government and businesses allows for consolidation of computing services, aggregation of financial services, etc. How does this influence where and how requirements are coordinated, and managed and how does it influence the interaction between local and global requirements? • Higher level software abstraction mechanisms like model driven architectures (MDA) or software as service (SaS) have emerged. How does one integrate these into requirements discovery and validation processes? How do they influence the scope and evolution of requirements and their discovery? • How do new information architectures force conformity to requirement and related rules sets? Is this is good thing and under what circumstances? • Tradeoff's are an important element of design: how do you design for tradeoffs when new technologies and anticipatory standards emerge? • Can we analyze how requirements are learned and diffuse along with new technologies? • Can we understand how new standards shape and interact with requirements?