The brave new world of design requirements☆

Matthias Jarke a, Pericles Loucopoulos b, Kalle Lyytinen c, John Mylopoulos d, William Robinson e

a RWTH Aachen University, Germany
b University of Loughborough, United Kingdom
c Case Western Reserve University, United States
d University of Trento, Italy
e Georgia State University, United States

A B S T R A C T

Despite its success over the last 30 years, the field of Requirements Engineering (RE) is still experiencing fundamental problems that indicate a need for a change of focus to better ground its research on issues underpinning current practices. We posit that these practices have changed significantly in recent years. To this end we explore changes in software system operational environments, targets, and the process of RE. Our explorations include a field study, as well as two workshops that brought together experts from academia and industry. We recognize that these changes influence the nature of central RE research questions. We identify four new principles that underlie contemporary requirements processes, namely: (1) intertwining of requirements with implementation and organizational contexts, (2) dynamic evolution of requirements, (3) emergence of architectures as a critical stabilizing force, and (4) need to recognize unprecedented levels of design complexity. We recommend a re-focus of RE research based on a review and analysis of these four principles, and identify several theoretical and practical implications that flow from this analysis.

© 2011 Published by Elsevier B.V.

1. Introduction

The genesis of Requirements Engineering (RE) research around the mid-1970s was motivated by practitioners, who noticed the urgent need for disciplined RE in software projects that had grown large and unmanageable [1,2]. Much of RE research since then has focused on artifacts that maintain the intellectual discipline by helping capture, share, represent, analyze, negotiate, and prioritize requirements as a basis for design decisions and interventions. This is evidenced by the volume and impact of a plethora of requirements-related papers published in top software engineering conferences and journals (for a survey see [3]. Due to its practical origins, it is not surprising that some of the findings of this research, such as the use of business and system modeling (e.g., entity-relationship and UML diagrams, use cases [4,5], risk-driven methodologies [6], structured requirements documents, and requirements tracing [7,8], have found their way into design practice [9].

Yet, the environment in which RE is practiced now has changed dramatically from the landscape of 35 years ago. Partly, this is due to advances in hardware and telecommunications that has radically lowered computing cost and extended functionality. This has made software
ubiquitous in all walks-of-life. For example, for a large part of the population, smart phones and netbooks rather than workstations and PCs are becoming the computing device of choice; RFID tags and other sensors bring real-world measurements into the administrative world of Enterprise Resource Planning (ERP); and highly reduced communication cost moves high-performance computing and data services into “the cloud”. Partly, this is due to changes in task and organizational environments where software is either produced or deployed. Therefore, the field’s focus and scope has shifted from engineering of individual systems and components towards the generation, adaptation and maintenance of software-intensive ecosystems consisting of software, hardware, human and organizational agents, business processes and more. For example, large infrastructure players such as IBM, SAP, Oracle, Microsoft, or Google are jockeying to dominate with their platforms by offering software development services that permit order-of-magnitude productivity increases at the expense of vendor lock-in. This happens, in part competing with, and, in part cooperating with the Open Source movement. Within user organizations, the rapid globalization and re-organization of industries widens the boundaries of organizational information systems to inter-organizational systems where connectivity and semantic standardization become highly important. In contrast, many software products in the Web 2.0 ecosystems to inter-organizational systems where connectivity and semantic standardization become highly important. In contrast, many software products in the Web 2.0 world measurements into the administrative world of Enterprise Resource Planning (ERP); and highly reduced communication cost moves high-performance computing and data services into “the cloud”. Partly, this is due to changes in task and organizational environments where software is either produced or deployed. Therefore, the field’s focus and scope has shifted from engineering of individual systems and components towards the generation, adaptation and maintenance of software-intensive ecosystems consisting of software, hardware, human and organizational agents, business processes and more. For example, large infrastructure players such as IBM, SAP, Oracle, Microsoft, or Google are jockeying to dominate with their platforms by offering software development services that permit order-of-magnitude productivity increases at the expense of vendor lock-in. This happens, in part competing with, and, in part cooperating with the Open Source movement. Within user organizations, the rapid globalization and re-organization of industries widens the boundaries of organizational information systems to inter-organizational systems where connectivity and semantic standardization become highly important. In contrast, many software products in the Web 2.0 sphere are in a perpetual state of running beta versions [10], as time-to-market is more important than product quality. For all parties concerned, security has become a true concern for all ecologies.

Accordingly, a term design requirements rather than software requirements as an inclusive term better characterizes all requirements issues within these ecosystems that need to be addressed at the crossroads of business development, software engineering, and industrial design. This shift has created a strong need to re-think the fundamentals of RE research and to re-align them with the new RE practices. In this quest, both academia and industry need to seek to understand more deeply issues that underlie current RE and address associated challenges. We posit that answers to this challenge cannot come just from doing more of the same—i.e., traditional RE research focusing primarily on notations, tools, and methods. The RE field needs to carefully evaluate some of its sacred assumptions and as a result its research scope may have to become more interdisciplinary. This essay aims to identify some of these challenges based on a review of current state of the art and a detailed field and content analysis of extensive expert discussions and feedback on current RE practices. As a result we put forward four key principles that characterize future requirements engineering efforts and discuss their differences to current practices. We also outline main elements of a research agenda that is needed to address future RE challenges.

The remainder of the paper is organized as follows. We first offer a short synopsis of the history and state-of-the-art of RE research. Section 3 clarifies the data collection and analysis and the nature of data that shaped discussions around new requirements challenges as reported here. In Section 4, we discuss the changing nature of requirements condensed into nine issues as derived from empirical analysis. Section 5 forms the apex of our argument and proposes four new principles that underlie contemporary and future requirements processes. We conclude with a comparison with current RE practices (Section 6), and discuss implications of our deliberations for research and practice (Section 7).

2. Requirements research—a brief overview

Over three decades ago researchers and practitioners alike recognized the critical role of the requirements contributing to the eventual success (or not) of software development [1,2]. Requirements challenges were seen as inherent and risky elements for any successful system design effort, because of the complexity of the design task, the limits to human information processing, and the intricate interaction between designers and intended users [12]. In addition, the RE activity was found to be sensitive to other forces that shape organizational life in that RE is unavoidably intertwined with the politics of resource allocation and the legitimacy of decision-making related to system development [13].

In light of the myriad observed and important shortcomings tied to requirements processes, RE research acquired soon a significant visibility in software and system development research. In order to address observed shortcoming in RE practices, RE research has also remained predominantly prescriptive (see e.g., [1,2]). It is replete with analytical frameworks, standards for requirements quality, elicitation protocols, and modeling methodologies. A wide array of textbooks and reviews has been published on the most advisable (effective?) approaches to requirements engineering [14–21]. Moreover, these prescriptive methodologies have seldom been subjected to rigorous empirical scrutiny [22]. By comparison, a relatively small percentage of the literature has focused on advancing a deeper theoretical or empirical understanding of the nature of system requirements; how design requirements are discovered, defined, negotiated, and managed within organizations; or why these processes are so difficult.

It is important to note, however, that requirements processes are far from monolithic. Just as requirements represent one facet of a broader system development effort, so too requirements processes can be divided into a number of facets. Within the RE literature, multiple frameworks have been developed, positing anywhere from two to seven primary requirements tasks [23]. For the current review, we adopt a widely employed and straightforward categorization of the requirements processes into three facets: (1) discovery, (2) specification, and (3) validation and verification [17]. During discovery, designers develop an understanding of the application domain and infer specific design needs through
consultation with stakeholders and other sources [18].

Requirements specification is both a noun and a verb in RE parlance. As a noun, a specification is the document in which the requirements for a design effort are articulated thus representing an agreement between the stakeholders and the design team [16,24]. The verb suggests the process of developing and managing the specification document; it is the process by which the design team acquires, abstracts, and represents the requirements [17,22]. Finally, during requirements validation and verification designers ensure that the requirements are (1) of high quality, (2) by addressing the users' needs, (3) and being appropriate for the design effort, and (4) having no inconsistencies or defects [25]. While this tripartite characterization appears to imply a linear progression, the three facets are normally employed iteratively while progressing to more detailed levels [23]. Despite the strong interconnectedness of the facets, most requirements research has focused on only one at a time. Accordingly, a more detailed exploration of persistent and new challenges associated with each of these facets is warranted.

Discovery: discovery is the first task of any design effort as the development team must determine what organizational or customer needs must be addressed by the design artifact [17,26,27]. Requirements researchers have also stated that requirements are properly concerned with what is to be achieved by a design artifact, without regard to how it will be designed and implemented [14]. This argument suggests that the bailiwick of the developers (i.e., the “how”) is something distinct from the requirements themselves, and “leaves unresolved the question of whether or not it is possible or desirable to separate the ‘what’ from the ‘how’ in practice” [28]. The discovery is also often referred to as requirements elicitation. This conveys a belief sometimes held widely in the RE community that requirements knowledge is banked in users or other stakeholder’s heads, and must just be “teased” out by the designer through some techniques. Through discovery designers can also gain knowledge of the relevant application domain. As Loucopoulos et al. [17] note, understanding of the application domain “cannot easily be overestimated ... when you have to solve somebody else’s problem the first thing you have to do is to find out more about it” (p. 21; emphasis in original). This illustrates the assumption that the designer is in most cases regarded as an outsider in the application domain, who is brought in for a limited period to resolve a problem that is of potential concern to other stakeholders.

Traditional approaches to discovery cover a wide range of techniques [26,29]. Among the most widely ones are one-on-one interviews between a designer and a stakeholder, focus group discussions, and direct observations of business tasks [29,30]. A number of more intensive discovery techniques, such as protocol analysis [31,32] and the use of ethnography [26,33,34], have been proposed to render explicit tacit user knowledge. Finally, prototyping has been widely deployed as a way to expand requirements elicitation activities with concrete user experience and learning [35] and to establish a common understanding of the design idea [36].

Specification: as stakeholder needs and understandings converge, they must be rendered in some representational scheme or model. This rendering is typically referred to as the specification process. Perhaps no single subject within the RE literature has received more attention than that of modeling [37] to the extent that some argue that model development lies at the core of RE [38]. There are several benefits that have been attributed to the use of formal specifications. By encapsulating large amounts of information, requirements models establish baseline of designers’ understanding of a complex system. In addition, they (may) facilitate communication between distinct stakeholder groups [38]. Models also enable analysis to identify unstated requirements, predict behavior, determine inconsistencies, and check for requirements accuracy. Finally, models simplify and organize application domain knowledge by featuring its essential features through abstraction and decomposition. While each of the proposed benefits of modeling is sound in itself, these arguments also unpack widely held assumptions underlying modeling literature—an emphasis on designer’s perspective and the conjecture that the subject world will not change i.e., the model remains valid once established.

Within the RE literature, the focus has been squarely placed on ways in which modeling can enhance designer’s work with less regard for the other stakeholder’s preferences. Yet, most other stakeholders prefer natural language to more formal specifications [39] though researchers have long argued that the informal treatment of specifications leads to ambiguity, incompleteness, and inaccuracy [40]. Most treatments within RE research address therefore ways to overcome limitations of natural language and have pursued attempts to automate the generation of formal models from natural language [41–43]. Another stream has focused on ways to augment natural language representations with formal and semi-formal models of requirements [1,2].

Although specifications are typically articulated at multiple levels of detail for example at e.g., enterprise, functional, and non-functional levels [17], the bulk of the modeling literature focuses on modeling functional requirements at the system level by explicitly representing functional requirements about the design artifact itself by abstracting it from the environment and making it amenable to design. Over the last three decades a plethora of techniques has been advanced including the structured analysis and design techniques [2,44], statecharts [45], Petri nets [46], as well as object-oriented models [47]. While most of these modeling methods were introduced as separate techniques for representing an application domain, recent trends has been toward integrating design knowledge across them [17,48]. Recent work has also integrated enterprise level and non-functional requirements [17,48].

Validation and verification (V&V) addresses the question of whether or not the requirements processes have been conducted effectively and the degree to which the specification will support a productive development effort. There is an important nuance to note between the two V’s. Validation is the effort to ensure that requirements reflect accurately the
intentions of the stakeholders [49]. Verification, in contrast, focuses on the degree to which requirements conform to technical or formal standards of requirements quality [25,50] captures this distinction succinctly by mimicking Drucker’s famous quote: validation addresses the question “Am I building the right product?”; while verification asks “Am I building the product right?” [50], p. 75).

V&V is typically positioned in process descriptions of RE at the end of requirements process. In practice, V&V begins almost simultaneously with discovery and continues through the specification activity. When a designer uses paraphrasing to check his or her understanding of a stakeholder’s statement, validation is already taking place. Indeed, one of the primary approaches to discovery – namely prototyping – is therefore also identified as a validation technique [17]. In addition, V&V embodies much of the negotiation activity involved in requirements effort as developers seek to identify and reconcile disagreements between individuals or stakeholder groups [37,51–54].

The V&V effort inherits several of the assumptions associated with discovery. Most notably, V&V (and particularly, validation) reflects the belief that requirements exist in some perfect form in Platonic world within the heads of the users or other stakeholders. In this sense, validation maps the documented requirements to their original source and seeks to minimize the gap between the two and the design team is perceived to be a disinterested observer of the correctness of this mapping. Indeed, this separation of interests is reflected in actions that pervade the V&V process. Project sponsors and other stakeholders at expected sign-off on a design validated specification, and they will ultimately “own” the system created based on validated requirements.

Although this is a cursory treatment of RE research necessitated by a concern to brevity, our review highlights some of the key assumptions reflected in the prevailing RE research paradigm. The research discourse has maintained a focus on processes for the development of a single system and conceived it in isolation from the broader socio-technical ecology within an organization. In addition, we observe a distinctly designer-centric preference for formal notation techniques to document a project’s functional requirements (with the consequent disparagement of natural language). Perhaps most fundamentally, we see that requirements have consistently been characterized as residing outside of the developers themselves and remain fixed for the development and evolution of the system. We will next investigate to what extent these assumptions are met with the current practice.

3. Methodology

RE practices and principles were investigated through a field study[4], as well as in two workshops bringing together experts from industry and academia [9,55]. We used a rigorous empirically grounded research methodology that involved intense data collection in the field followed by a solicitation and integration of expert opinions through workshops and focus groups. We first conducted an extensive 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 RE challenges [4]. The study followed a similar approach as in [56]. The study sought participation from senior technology leaders within a range of Fortune 500 organizations and leading software developers. 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 studied systems and their requirements included the following:

- **Large, complex organizational information systems**: these systems, often supporting inter-organizational information exchange; included transportation systems, distribution networks, and logistics in defense.
- **Embedded systems**: these systems were integrated with complex mechanical or physical artifacts including automotive components or aerospace design environments.
- **eBusiness applications**: these systems were intended for use within Web-based delivery channels included portals, e-commerce platforms, and other Internet-oriented services.
- **Middleware systems**: these software platforms typically supported the exchange of data (either control or application) between distinct applications through messaging pipes.

The systems development efforts poured into these systems ranged from tens to hundreds of man-years and the project time frames varied from 6 months to nearly 10 years. The development costs ranged from several million to hundreds of millions of dollars.

Semi-structured interviews were conducted with 39 individuals on new challenges related to RE practices and design associated with these systems. The interviews ranged from 1 to 4 h in length. The interviews were recorded and transcribed to support subsequent formal content analysis of the data using technique recognized as grounded theory [57]. We used Atlas.ti software for coding and classification. Several cycles of coding resulted in a review of common requirements practices and nine requirements issues [4].

Two workshops followed the field study and they used the field study report as one input in addition to position papers written by the participants. The workshops brought together diverse experts from industry and academia working on RE topics [9,55]. Each workshop focused on a select set of issues based on the evolution of the dialog and solicitation of key findings as the debate unfolded. The workshops were designed to accomplish three objectives: (1) engage separate research communities in a dialog on requirements challenges and effective practices, (2) strengthen design science principles, and (3) open new vistas for the research on design requirements. The focused discussions in the workshops were recorded in workshop Wikis (see e.g., http://weatherhead.case.edu/requirements) and offered additional data set for further analysis.

Please cite this article as: M. Jarke, et al., The brave new world of design requirements, Informat. Systems (2011), doi:10.1016/j.is.2011.04.003
The first workshop was held in the United States in June 2007 [9], while the second workshop was held in October 2008 in Germany (http://drops.dagstuhl.de/portals/index.php?semnr=08412). In this workshop the participants were in particular asked to address or raise issues related to field study findings (as organized into nine issues shown in Table 1), or point out omitted issues in the empirical findings or extant research. The following issues were selected for deeper exploration in focus groups:

1. Analyses of contemporary practices in the form of case studies, field studies, and experience reports of successful or failed practices that would expand understanding of current practices.
2. Based on (1) identification of emerging critical issues in the design of complex software-intensive systems.
3. Based on (2) strengths and weaknesses of earlier requirements and design research in different disciplines.
4. Opportunities for intellectual cross-fertilization across disciplines in design and requirements based on (2) and (3).
5. Identification of new avenues in requirements research by mapping out a new landscape and emergent challenges based on (4).
6. Identification of new theories and research methodologies pertinent to the emerging challenges based on 5.

In particular, participants were asked to reflect on their research and practical experiences in a set of chosen issues and identify and rank areas that would deserve greater attention. As a result participants identified, presented and explored several new research themes which were then refined and solicited into a list of issues and questions. As a next step the organizers (authors of this paper) wrote using this workshop material as input a white paper on RE research challenges and principles [9,55,58]. This “manifesto” was distributed to the participants of the second Workshop as a main input. As a reflection to this input the second workshop identified new challenges and crafted principles discussed in the white paper in more depth. These included in particular:

1. Continuous interlinking of multiple concepts of design, and its intertwining with requirements.
2. Evolution and management of requirements under growing complexity.
3. Architectural implications and platform strategies.
4. Identification of changing stakeholder roles and management challenges in handling stakeholder arguments around RE.

In consequence the second workshop identified new implications for research and practice such as the growing importance of the edge, new capability-based organization forms and the role of business semantics as discussed below. We will next probe the changing nature of requirement as observed through a field study and expert discussions.

4. The changing nature of requirements

The “requirements mess” has remained a pernicious challenge and motivation over the last 30 years for RE scholars. In addition, it does not show any signs of disappearing [59]. RE researchers have been persistent to note that the leading sources of project difficulty – lack of user input, incomplete requirements, and changing specifications – are directly related to flaws in design requirements [37,60–63]. Yet, as observed in the field study [9] most large projects follow many of the accepted canons of good RE: requirements and design modeling, requirements reviews, and risk driven development. In this regard, it appears that effective management of design requirements demands now more. One reason is that the RE environment now involves elements and behaviors that were not present 30, or even 20 years ago [56]. Its organizational environment, target technologies, processes,

<table>
<thead>
<tr>
<th>Critical requirements issues</th>
<th>Brief description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Target platform</strong></td>
<td>Requirements focus simultaneously on business processes, and requirements for technological artifacts driven by these processes</td>
</tr>
<tr>
<td>Business process focus</td>
<td>Requirements involve the demand for a seamless user experience across applications</td>
</tr>
<tr>
<td>Systems transparency</td>
<td>Requirements focus on integrating applications, rather than development of new ones (i.e., less green-field development)</td>
</tr>
<tr>
<td>Integration focus</td>
<td>Purchase of COTS software rather than internal development. This has led to market-driven vendor-led requirements and knowledge brokering</td>
</tr>
<tr>
<td>Packaged software</td>
<td>In addition to increasingly diverse stakeholders, requirements processes are distributed globally across multiple organizations, groups and social worlds</td>
</tr>
<tr>
<td>Development process</td>
<td>Architectural considerations and associated evolutionary paths take a central role and drive business, product and application requirements</td>
</tr>
<tr>
<td>Distributed requirements</td>
<td>Requirements need to be iteratively developed across multiple levels of abstraction, design focus, or temporal horizon</td>
</tr>
<tr>
<td>Centrality of architecture</td>
<td>While some forms of design complexity have been reduced (e.g., loosely coupled components), the overall interaction complexity of the design ecology has risen enormously</td>
</tr>
<tr>
<td>Layers of requirements</td>
<td>Requirements processes must accommodate the need for continued evolution of problems and artifact solutions after initial implementation</td>
</tr>
<tr>
<td>Interdependent complexity</td>
<td></td>
</tr>
<tr>
<td>Fluidity of design</td>
<td></td>
</tr>
</tbody>
</table>

Please cite this article as: M. Jarke, et al., The brave new world of design requirements, Informat. Systems (2011). doi:10.1016/j.is.2011.04.003
and fundamental problems have undergone a tectonic shift. RE practices of today are therefore imbued with novel challenges and fresh opportunities [9].

First, the economics of RE has changed. Large systems, such as ERP systems, need more rigorous return-on-investment (ROI) analysis, but at the same time horizons for ROI have shrunk to 18–20 months thanks to massive reuse and commercial-off-the-shelf (COTS) software deployment. Second, there is practically no green-field software development; RE acts more like the Roman god of gates – Janus – with one face looking at new business and technological challenges and opportunities, while another face is gazing at existing (technological, organizational, social, and political) environments and resources. Third, the scaling towards software-intensive ecosystems has resulted in exceedingly complex and non-linear dynamic dependencies between ecosystem elements—“green IT” being just one of the latest buzzwords 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 search for new design trade-offs between efficiency, openness, and flexibility.

These trends have been compounded by outsourcing and off-shoring, which demand disciplined evolution and management of explicit specifications as a basis for delegation and framing of design problems. Fifth, RE now cuts across industrial design (e.g., pervasive and ubiquitous computing), media design (e.g., e-commerce and media applications), interaction design (e.g., new modalities of interaction in mobile computing, telematics, etc.), business process design (e.g., open business platforms), enterprise architectures, and regulatory and juridical issues (e.g., corporate governance, intellectual property management). Overall, design requirements need to capture and coordinate increasingly diverging and dynamic needs of heterogeneous stakeholders during the evolution of software based product viewed often as a service, or a platform. What are the critical issues that emerge during RE in this brave new world?

Table 1 shows nine critical issues that were solicited in the field study[4]. These are divided further into the changing nature of the object of RE (target platform), and the process of RE (development process). Overall, these issues resonate well with the debate Simon engages in his design classic, The Sciences of the Artificial [64]. On the one hand, software designs resemble increasingly continuous and dynamic searches for satisfying solutions—not an optimized and fixed solution at one time point conforming to fixed requirements as assumed in the past. On the other hand, they go beyond Simon’s original model in that they emphasize sense-making in shifting and complex environments [65], and associated continuous problem framing over problem solving in a bounded context thus contesting the assumption that designer is an outsider or the subject world remains invariant. To wit, these changes are changing the three classic RE problems that underlie past RE research as follows:

First, the design requirements problem succinctly pointed out by [1,2] can now be stated as follows: what is the emergent behavior and dynamics of the software artifact and its environment in their evolutionary trajectory? Now users, designers and other stakeholders need to ask: will the system continue to satisfy our emergent goals, and what those goals could be expected to do during the artifact’s lifetime; in contrast to the older problem: what are the (fixed) goals of the system and what is it expected to do?

Second, the specification problem can 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 emerging environments and the notations used to represent and predict it? Accordingly, designers need to ask how they can represent, communicate and analyze increasingly complex and dynamic systems and their emergent requirements, and how this is possibly condition by the nature of presentations brought to bear in the design context in contrast to the older problem: how to faithfully represent the system components, their relationships and behaviors in ways that guarantee that these meet functional and non-functional requirements?

Third, the predictability problem of designs can be stated as follows: how does the artifact and its behavior change the environment as to make our predictions of system behaviors faithful? In other words, now designers need to attend more closely to the continuous dynamic composition of the system and its environment, and how do they together differ from the environment in separation. Designers need to predict faithfully the impact of the system on the environment, and vise versa. This is a different problem from those faced earlier where the system was assumed to not affect the environment, or the environment the system, with rare exceptions (e.g., [65]).

5. Four requirements principles

As mentioned in Section 2 past RE research has been informed by several principles that served well addressing the classic requirements problems. These included: (1) separation of concerns (notably, the ‘what’ from the ‘why’ and the ‘how’), (2) information hiding, and (3) the principle of abstraction (see e.g., [1,2]). These principles reduced design complexity given the requirements and goals by localizing design decisions. They helped reduce the mutual interference of decisions and supported both the analysis and prediction of system behaviors and structure from specific viewpoints. This helped solve the design requirements, the specification and the predictability problems.

What are the new key principles that will underlie successful design and RE in the “brave” new world of requirements? What new principles will help us address in the new context the design requirements problem, the specification problem, and the predictability problem? We propose four principles that were derived from our review of expert opinions and related discussions in the workshops, our ongoing discussions how to handle them and a review of broader literature in several field including architecture, complexity theory and economics dealing with design dynamism and complexity. These four principles are as follows:

1. Intertwine requirements and contexts: the necessity to consider the relationship between requirements with
their organizational/business context, in addition to their relationship with their implementations.

Rationale: this principle recognizes the design requirements problem and the demand for understanding evolutionary trajectories.

2. Evolve designs and ecologies: the necessity to view design and design processes as evolving elements in an ecology.

Rationale: this principle recognizes the new design requirements and specification problems and the increasing demands they pose to analyze evolutionary principles of a set of heterogeneous elements comprising the requirements space.

3. Manage through architectures: the necessity to consider architectures as enablers and constraints in the continuous creation and shaping of design ecologies.

Rationale: this principle recognizes the specification problem and the shifting focus towards ecologies where emphasis must be placed on antecedent “shaping” or “generative” factors that affect the organization and evolution of the ecology.

4. Recognize complexity: the necessity to consider the heightened interaction complexity of requirements processes which demands new ways to approach design problems and manage requirements.

Rationale: this principle recognizes the predictability problems and that the new interaction complexity requires designers to heed on the external relationships of the software and their evolution as reflected in the requirements and related need to enhance adaptability and evolution of the software system.

We next describe each principle in more detail. We illustrate the content of each principle, the new research questions it sparks, and discuss some of the emerging research to address the principle.

5.1. Intertwine requirements and contexts

The debate about the role of requirements is as old as the field itself. While a rough consensus has been reached that requirement constitute a conceptual – but not necessarily temporal – prerequisite for downstream development, capturing the problem to which design and implementation constitute a solution. At the same time, there is a great deal of controversy on how problem and solution spaces interact during the evolution. One school of thought regards the influence of implementation on requirements as being harmful [66]. 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. This school regards requirements as the downward bridge between the subject and system worlds by claiming 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 [67].

The review of existing practice [4] shows that implementation and its requirements specification are now intrinsically intertwined. In fact, many requirements emerge from existing solution spaces. Accordingly, the concept needs to be extended to the whole system context. In addition, the salient factors shaping RE seem to be innovation and effective differentiation. The interplay between the two worlds has thus become more intricate, complex, dynamic, and generative. In these innovation-driven settings, requirements become part of both the business solution and the system solution, and they constantly bridge 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 organizational innovation. Therefore, software requirements need to be dynamically situated between these spaces as they intertwine organizational and implementation considerations, providing leverage to influence both.

The constant intertwining of some systems and their ecologies may face a practical world-model limit. 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 changing needs. The level of stability of the world assumptions is less limited in context-aware, customer-focused applications. The unavoidable intertwining between requirements and contexts will make designers constantly seek correspondence between the models within a system and its world context. Only systems embedding an adequate, flexible, and evolvable world-model are likely to survive. The idea of evolutionary systems and variability generation and selection principles aims to meet this need, in part. However, little attention has been given to the challenge of formulating evolutionary world models that form the basis for the necessarily simplifying, but evolving model assumptions in the system. Thus, designers must monitor and evolve their understandings of the world, and sustain an adequate correspondence between the world and its system models.

Overall, RE processes face new kinds of uncertainty that goes beyond traditional RE uncertainty characterized by (1) requirements identity (knowing requirements), (2) requirements volatility, and (3) requirements complexity [68]. In addition, two new kinds of uncertainty are now prominent: (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. These two new levels of uncertainty highlight the need for increased
run-time monitoring to maintain the fidelity of the world-model intertwining with requirements, which again shows the intricate and dynamic intertwining of the problem and solution spaces [69].

5.2. Evolve designs and ecologies

Meeting stakeholder needs is the fundamental axiom of any RE endeavor. As requirements increasingly intertwine with organizational and implementation concerns, they will constantly and non-linearly evolve as part of an "ecology". Traditionally, causes of software evolution have been classified into: (1) software-related, (2) documentation-related, (3) properties of the software, and (4) customer-experienced functionality [70]. Evolution has been studied mainly as a software design problem and it has been addressed by improving methodological support (e.g., how can development activities most effectively incorporate evolution?) and its management (e.g., how can one record and trace software releases, or link code to changing domain knowledge?). Now, we need to address the reality of an ever incomplete and evolving design by asking: what are the principles that guide developers evolve incomplete designs so that they remain functionally adequate, but offer flexibility-in-evolution? What are appropriate co-evolutionary design methods to achieve these goals? How does one determine the impact of co-evolutionary design change?

Activities in open source development, such as inter-project merging and the creation of new software artifacts, for example, compound the need for new frameworks to cope with requirements evolution. Another example is agile methods and scenario-based modeling which offer a means to better cope with the fast paced evolution of requirements ecologies [71]. Similarly, research into co-evolution and co-design [72] has addressed drivers and interaction laws that deal with the intertwining of contexts and requirements. Yet, such studies are in their early stages, and agile methods only deal with micro-level evolution of local tasks, but ignore their recursive nature as the change propagates across higher-level architectures and systems. We need longitudinal studies of the dynamics of software ecologies and how different causes ranging from technological, user-level learning, organizational policies, market-based, and regulatory changes intertwine and generate new evolutionary paths and how such paths vary depending on the history of the system and its main structural elements.

5.3. Managing through architectures

Architectures are blueprints that connect high level organizational, business, or implementation elements and concerns with a long-term evolutionary perspective. Organizations now increasingly seek to conform to enterprise, business or information architectures that provide stability, scalability and control over their data, business processes, and decision procedures [73]. Designers have also relied on implementation architectures while evolving their designs. In its variety of forms, an architecture provides the stepping-stone necessary to understand and evolve any system functionality across and within domains. Through release planning, requirements play a central role in system evolution, whereas architectures provide "nearly" fixed points of reference to moderate, constrain, and enable requirements evolution [64]. Such dependence on architecture is inherent e.g., in Lehman's law that "the incremental growth (growth rate trend) of evolutionary software systems is constrained by the need to maintain familiarity" [74]. Architectural dependencies arise also in approaches such as IKWISI- (I’ll Know It When I See It), and COTS-based software deployment [75].

Although RE research has paid significant attention to software architectures, it offers limited insight into the role of architectures in the new RE terrains. Many of the past studies have focused on organizing sets of design elements and their components in the context of a single system. Accordingly, they view architectural design as akin to generating a blueprint for a single house. In the context of dynamic and more complex software ecologies as stated by three new requirements problems, such analog fails. In the brave new world, managing through architectures is mainly about simultaneously generating and evaluating multiple and multifaceted plans analogous to urban planning. Like urban planning blueprints, architectural models provide the key artifact for coordinating components, functionalities, and their evolution. As in urban planning, the architectures in RE may have alternative and overlaying variation points that influence the evolution of the software ecology. As in urban planning, architectures embody specific business models or design visions, and come in different forms in different design contexts. By doing so, they integrate the needs of multiple stakeholder groups with varying roles. Finally, like urban plans they involve a high degree of complexity and interdependencies. Therefore, in the new RE, we see an increasing need to understand the variation between types of architectural models and how they relate to specific system families, their ecologies, and evolution.

Recent attempts to deal with architectural considerations include the transfer of industrial concepts, such as product lines to the software field [76]. These help manage and cope with continuous and rapid change in software by defining the scale, scope, and direction of its variance, selection, and retention. Other architectural models such as business architectures [72] help stakeholders to envision the impact of proposed changes on business by providing contextual information that allows for selecting variation points for requirements that help coordinate requirement sets across multiple stakeholders. Yet, many research challenges remain in exploiting the advantages of the notion of architecture: how do architectures influence the evolution of requirements and their identification? What is the nature of requirements discovery and elicitation under varying architectural principles? Is it possible or even desirable to construct a single common ontology of business, information, and technology architectures? How does one relate different architectural presentations and reason around them? How can architectures help in the dynamic composition of systems and ecologies?
Complexity fundamentally arises from the existence of multiple uncertain futures that relate to software systems and their evolving ecologies [77]. In this sense the complexity we face is common to the concerns of complexity that have been the center of organization and policy scientists since the 1960s [78]. In short, the software design is changing of just being complicated to being inherently complex [79]. In such systems, the behavior of the overall system is surprising and hard to predict (the predictability problem) because it is non-linear. A mix of human, social, political, economical, technological, and organizational factors and their constant interactions has a bearing on the level of complexity associated with RE making the behavior of the system and its environment non-linear. Overall, a new kind of interaction or systemic complexity\footnote{This should not be confused with computational complexity as defined by well-known complexity notions, such as NP hard problems.} needs to be reckoned and managed, where design amounts to problem solving and framing with inherent uncertainties and feedback driven by partial unknowns [80]. In such systems, intervening by adjusting one or two parameters related to requirements can drastically change the behavior of the whole system, and the whole is often very different from the sum of its parts. Dealing with such design complexity impacts two areas of RE: (1) strategic decision-making in generating and selecting requirements and how such a process can become self-organizing, as well as understanding its impact on the ecology; and (2) selecting tractable design-approaches that make complex system designs possible in which the rate of change is matched by the system designers and where the stability and the change are constantly “matched” [81]. The former is concerned with how to relate and manage complexity with stakeholders within their subject world, whereas the latter influences behaviors within the development world.

In the brave new world of RE, the implementation impacts not only the technical system, but also their organizational and social settings and often increases interaction complexity in unpredictable ways. In addition, the increased variety of requirements that emanate from diverse communities need to be negotiated, evaluated, and selected, thereby compounding complexity. Qualitative and often structural conceptual models, such as goal or business models, while rich and useful in representation and analysis for design are less helpful for stakeholder evaluation and understanding the interaction complexity [82]. Due to its inherent complexity, it is also difficult for some stakeholders to visualize a design and infer from it a system’s behavior. For designers, it is tempting to think that stakeholders understand a design, when they do not really understand it at all. Therefore, many questions remain poorly understood concerning design complexity: what is the nature of design complexity and increased interaction complexity, and how can we identify, analyze and measure it?

One way of coping with design complexity is through simpler architectural designs and control that allow nearly decomposable system designs. This mitigates complexity by ensuring that interactions among components are weak, though not negligible and overall designs remain simpler while their interactions are capable of generating high levels of complexity [78]. 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. Requirements methodologies need to be formulated based on simpler principles of self-organizing systems that enable rapid change by allowing constant and natural re-organization of the design space [78,81]. In this sense, an emergent design methodology that self-organizes will be an improvement over conventional \textit{a priori} methodologies. For example, open source systems following nontraditional methodologies can evolve systems faster than traditional lifecycle and requirements-driven development approaches due to this constant process of self-organizing using simple rules of decomposability [83]. The co-design and co-evolution of a system and its ecosystem plays a pivotal role in shaping the design complexity, as does the 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” [83].

Overall, a better grasp of complexity can be obtained, if system descriptions are tested against local concepts familiar to stakeholders, and multiple scenarios are played out by fixing key parameters as proposed by the architecture. Experimenting with different scenarios using feedback has proven a powerful means for discovering and refining requirements with heightened complexity [84]. Still, we need to examine: 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?

5.5. Our key design principles: summary

The four principles are shown in Table 2 together with the rationale and motivations for deploying each principle. We do not claim that each principle applies in all contemporary design contexts. Still, a great deal of design problems can be characterized by a single system, smaller design and solution spaces, fixed goals and linear change. Here, the received principles of separation of concerns, information hiding etc should still work.

When designs become increasingly fluid, however, i.e., when goals and stakeholders start shift, when software designs integrate or embed with novel business processes, or when designs involve significant distribution of requirements across subject and development worlds we can expect the principle of intertwining of requirements and contexts to become pivotal. In other words, factors as those identified above can serve as threshold indicators suggesting that while abstraction still is
Table 2
Four key requirements principles.

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intertwine requirements and contexts</td>
<td>Requirements are interdependent with their social and technical contexts. As boundary objects in the intersection of 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.</td>
<td>Intertwining between business, organizational, community context and requirements is as important as its counterpart between requirements and implementation.</td>
</tr>
<tr>
<td>Evolve designs and ecologies</td>
<td>Design ideas and artifacts evolve from stakeholder preferences to implementations. Evolution needs to be managed through selectively freezing some aspects while allowing increased variation, dynamic selection and retention of structures and behaviors in other aspects.</td>
<td>RE Problem addressed: design requirements problem, specification problem</td>
</tr>
<tr>
<td>Manage through architectures</td>
<td>Architecture is the least evolving and most widely referenced anchor of any design, be it a business, information, or implementation architecture.</td>
<td>RE Problem addressed: specification problem</td>
</tr>
<tr>
<td>Recognize and mitigate against design complexity</td>
<td>The necessity to consider simultaneously a large number of issues and their non-linear interactions during design behaviors in other aspects.</td>
<td>RE Problem addressed: predictability problem</td>
</tr>
</tbody>
</table>

The four principles exhibit a high degree of causality (Fig. 1): intertwining increases the need for co-evolution; co-evolution leads to increasing need to manage the evolutionary process with appropriate architectures; this, in turn, increases the degree of intertwining. This causal loop suggests that interactions are non-linear that will inevitably drive up complexity, unless other mitigating factors intervene as discussed above.

We can also ask whether these four principles are disjoint, and complete i.e., whether they overlap and whether there are additional principles. We have briefly addressed the question of disjointness by reviewing the distinct features and antecedents of each principle. This suggests that we cannot truly reduce the principles into a fewer number of principles. With regard to complexity we face the classical choice problem, which is often denoted as the model specification problem: do these principles account for the most new unaccounted “variance” in the way in which requirements are different from the past? Naturally, the time and constant falsifiability tests are the only sure way to address this problem. But based on the debates in the workshops and constant tests of connecting the principles to the current issues of RE practice we remain convinced that they at least offer a minimum specification of the new principles.

6. Comparisons with conventional RE practices

Conventional RE principles have been codified within established organizational and business models that underlie requirements practice. Table 3 shows one such model—CMMI requirements practices. ISO 9000 and IEEE Standard 830 contain similar, document-focused recommendations. Others have expanded these standards with guidelines for using specific requirements techniques (e.g., [18]. Such practices form the basis for assessing an organization’s requirements maturity [85]. Generally speaking, these practices are considered necessary, but perhaps not sufficient for effective RE practice [86].

In their 1993 status report, Hsia et al. enumerate several barriers to success in requirements practice including: unrealistic schedules, hidden requirements, communication barriers, over-reliance on CASE tools, insufficient RE knowledge, requirements change, and ambiguous relationships between requirements and object-oriented designs [39]. The last two are reflected in the new principles of intertwining and architecture. The idea of insufficient RE knowledge is reflected in the principle of complexity and co-evolution. The others seem
to be caused by the last three. Thus, we can argue that addressing and understanding the four new principles is paramount to building RE practices on a firm ground.

Two subsequent roadmap articles in 2000 [37,87] summarize several common practices (such as [18], but also raise important concerns that have not been successfully addressed: (1) modeling must encompass social, environmental, and architectural descriptions, in addition to software modeling; (2) incomplete and inconsistent models are the norm—consequently, their evolution must be monitored and managed; and (3) reference models (of architectures, COTS applications, domains, requirements) should be applied to new projects. It is these very concerns that are reflected in our new RE principles.

The new RE principles espoused here augment conventional RE principles when the nature of requirements involves high levels of dynamism and complexity (Section 2). As introduced in Section 1, RE research has focused primarily on notations and tools to manage internal complexity when the environment is relatively fixed, with less emphasis on software system ecologies. While addressing the ecology, the new RE principles focus more on software’s messy, real-world interface over time (principles 1 and 2 of Table 2). With this expanded focus comes increased complexity, which calls for greater and different kinds of management effort than conventional RE (principles 3 and 4 of Table 2). Thus, the new RE principles, should lead to new requirement management practices.

To clarify the implications of the new RE principles, Table 4 summarizes new requirements practices in the style of CMMI practices. The presented practices illustrate how the new principles could be supported and enacted in real world settings. The list presented in the table is not claimed to be a complete nor the most effective set of practices. By analyzing conventional RE principles, as

---

**Table 3**

Summary of conventional (CMMI) requirements practices.

<table>
<thead>
<tr>
<th>Requirements development</th>
<th>Requirements management</th>
</tr>
</thead>
<tbody>
<tr>
<td>SG 1—develop customer requirements</td>
<td>SG 1 manage requirements</td>
</tr>
<tr>
<td>• SP 1.1—elicit needs</td>
<td>• SP 1.1—obtain an understanding of requirements</td>
</tr>
<tr>
<td>• SP 1.2—develop customer requirements</td>
<td>• SP 1.2—obtain commitment to requirements</td>
</tr>
<tr>
<td>SG 2—develop product requirements</td>
<td>• SP 1.3—manage requirements changes</td>
</tr>
<tr>
<td>• SP 2.1—establish product and product component requirements</td>
<td>• SP 1.4—maintain bidirectional traceability of requirements</td>
</tr>
<tr>
<td>• SP 2.2—allocate product component requirements</td>
<td>• SP 1.5—identify inconsistencies between project work and requirements</td>
</tr>
<tr>
<td>• SP 2.3 identify interface Requirements</td>
<td></td>
</tr>
<tr>
<td>SG 3—analyze and validate requirements</td>
<td></td>
</tr>
<tr>
<td>• SP 3.1—establish operational concepts and scenarios</td>
<td></td>
</tr>
<tr>
<td>• SP 3.2—establish a definition of required functionality</td>
<td></td>
</tr>
<tr>
<td>• SP 3.3—analyze requirements</td>
<td></td>
</tr>
<tr>
<td>• SP 3.4—analyze requirements to achieve balance</td>
<td></td>
</tr>
<tr>
<td>• SP 3.5—validate requirements</td>
<td></td>
</tr>
</tbody>
</table>

---

Fig. 1. Interactions between four principles.
The idea of architectures residing at multiple levels and viewpoints both implies and limits requirements, which in turn can be used to coordinate and manage requirements evolution. In contrast, CMMI SG 2 assumes no specific role for architectures within their waterfall model: “Customer requirements are refined and elaborated to develop product and product component requirements.” Thus, new RE practices explicitly consider the influence and importance of architectures on requirements and their evolution (SG 3 of Table 4).

4. Recognize and mitigate against design complexity: New RE considers the design complexity to be inherent the requirements level due to the dynamism and interactions between the system and the environment. Traditionally, design complexity was part of effort estimation and project planning where a fixed level of complexity was assumed. In the new RE, design complexity rises to the level of other important requirements qualities, such as risk, which have a defined role in project management [89]. Thus, the new RE practices explicitly identify and analyze design complexity and ways of mitigating against it.

With recognition of the new RE principles will come new RE research directions in how to integrate these principles into effective RE approaches and practices, and how to improve RE methods and tools. In this regard, Table 4 hints at several fruitful new practices. We will next discuss the implications of these changes in broader context of RE research.

7. Implications for research and practice

Over its 30-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 dynamic change between the business environments and the design and implementation worlds. As Fred Brooks noted in Dagstuhl workshop: “Design is not about solving fixed problems; it is constant framing of solution spaces”. This evolution has now probably reached a new tipping point characterized...
by unprecedented scale, complexity, and dynamism as reflected in the four requirements principles. This new brave RE world calls for fresh ways to think about requirements and their role. As in earlier turning points, such as the software crisis in the 1970s, the new world demands a resolute and careful intellectual response.

The four principles have numerous implications for research and shaping of the future RE research agenda. In the course of the empirical research and the two workshops, a large number of specific research questions were identified and debated which offer a useful glimpse to contours of relevant future research on RE. They are

Table 5

<table>
<thead>
<tr>
<th>Topic</th>
<th>Topic description</th>
<th>Research questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Intertwine requirements and contexts: example business process context</td>
<td>Business process or related user 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.</td>
<td>• Can organization design anymore be separated from infrastructure and information system design? • How can we detect and deal with new types of indeterminism when dealing with a multitude of business rules? • Issues traditionally embedded into operating systems, gets on the organization level (scheduling, etc.). The power of process modeling is 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 data? • 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? How do you validate them? • How to model, analyze, and align business models and associated software models like e.g., software platforms?</td>
</tr>
<tr>
<td>1. Intertwine requirements and contexts: visualization, representation and analysis of requirements</td>
<td>Visualization and information re-presentation play a critical role in requirements processes. They shape requirements discovery, organization, validation and verification. New requirements features necessitate powerful capabilities that allow visualization of requirements and systems across stakeholders and design contexts.</td>
<td>• 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 one user works with many computers, in distributed environments, which serve multiple people. How can we represent these communities and their growth in our theorizing and analysis?</td>
</tr>
<tr>
<td>2. Evolve designs and ecologies</td>
<td>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. Novel 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 will become diverse and more powerful. This affects requirements negotiation as designers need to recognize critical killer/non-negotiable requirements.</td>
<td>• How do environments evolve designs as the environments constantly and increasingly change? • What changes in requirements are due to the introduction of new technological capabilities? • What is the maximum scope and rate of change that is sustainable? • What types of new “informalisms” we need to identify and make sense of the change, in addition to old formalisms? • What types of run time mechanisms we need to monitor and evaluate satisfaction of the requirements? • How can we formulate and choose good design metrics and create a “design dashboards” for fluid designs”? • Requirements come now often after implementation. How can we make sense of emerging webs of discourse that negotiate and make sense what are the directions of change with these requirements? • What are the consistent properties (constraints) i.e., the “design anchors” that help keep design stable? What can become the decision basis of sustainable incomplete designs?</td>
</tr>
</tbody>
</table>
Table 5 (continued)

<table>
<thead>
<tr>
<th>Topic</th>
<th>Topic description</th>
<th>Research questions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>• How to integrate fixed and soft requirements and best effort designs based on evolving preferences?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• How to adapt to new requirements or satisfy existing requirements through software based monitoring and evaluation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Can we analyze how requirements are learned and diffuse along with new technologies?</td>
</tr>
<tr>
<td>3. Managing through architectures</td>
<td>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.</td>
<td>• Architectural plans in enterprise wide systems allow 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?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Higher level software abstraction mechanisms like model driven architectures (MDA) or software as service (SaaS) 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?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• How do emerging standard data and process architectures force conformity to requirement and related rules sets? Is this is good thing and under what circumstances?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tradeoffs form an important element of design: how do you design for trade-offs when new technologies and anticipatory standards emerge?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Can we understand how anticipatory standards shape and interact with requirements?</td>
</tr>
<tr>
<td>4. Recognize and mitigate against design complexity: systems perspective</td>
<td>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.</td>
<td>• How can we address questions of centralized control vs. distributed information requirements in new ways as the complexity increases?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 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?</td>
</tr>
<tr>
<td>4. Recognize and mitigate against design complexity: stakeholder perspective</td>
<td>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.</td>
<td>• How can requirements processes and stakeholder involvement embrace both technical and political considerations at the same time and address simultaneously technical and political requirements?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• How do we navigate through design spaces when different stakeholders have a different understanding of terms and world assumptions?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Can different design artifacts be used to make some stakeholder believe that they have more shared understanding than actually prevails?</td>
</tr>
<tr>
<td>4. Recognize and mitigate against design complexity: conceptualizations of design</td>
<td>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.</td>
<td>• How to put multiple owners of problems “in charge” of their problems?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• How to explain or analyze adequately ill-defined problems, and how does externalizing some of them improve the design ability?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• How to design an ecology instead of a bounded system?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 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?</td>
</tr>
</tbody>
</table>

Please cite this article as: M. Jarke, et al., The brave new world of design requirements, Informat. Systems (2011), doi:10.1016/j.is.2011.04.003
offered here (in Table 5) as starting point for a continued debate for future research on RE though due to concerns of brevity they cannot be discussed here in all detail. Table 5 provides a list of research themes and issues in need for additional research, organized according to the four principles. Starting from the complexity principle, they regard both the complexity of systems and the related complexity of stakeholder settings, leading to fundamental questions also concerning the basic conceptualizations of design we should pursue. In information systems arena, the intertwining of requirements and contexts is perhaps most virulent concerning the business context, even though other context aspects such as locality, user competencies, or device contexts are gaining in importance as people are using increasingly embedded systems on the move. This will also change the visualization, representation, and analysis tools we need in requirements engineering. One crucial aspect not only here is the fluidity of designs, one of the key aspects of evolution. Last not least, architectures are hoped to be the relatively stable element in this continuously evolving world, but significant research questions remain how exactly they need to look like, and by which tools and properties they should be supported.

There are also multiple implications for RE practice: both at the management and at the engineering level. For the sake of brevity, we discuss below three closely related strategies that need to be pursued and are being exposed by the four principles.

**Importance of understanding the edge**: the criticality of understanding and creating the edge—where the boundaries of the system and its functionality are “shifting” is becoming highly critical in RE. This is at the heart of managing the dynamics of the specification and predictability problem. Due to these dynamics many of the old ‘wisdoms’ are disappearing. For example, the distinction between users and developers is vanishing: the user networks and the developer networks are becoming increasingly fuzzy and intermingled. Many contributors to designs and even implementations are no longer located in the development kernel, but come from the shifting network edge. This situation generates greater diversity of system variants and system uses. Accordingly, web-based service networks (e.g., mashups) have proven to be the infrastructure of choice for such settings, offering quick and dirty design and validation of changes targeted at limited market portions. At the edge, design incentives now often go beyond monetary ones. In contrast, *bricoleurs* at the edge must be harnessed to contribute to usability and functionality through multiple related strategies and organization. This is broadening often design environments towards Open Innovation, which involve huge numbers of customers in RE organized in a growing social and technology network. For example, the success of Facebook as a leading social network is largely due to its strategic decisions to open up its APIs and their requirements evolution for third-party developers in 2007 [90]. Thus, transparency, accountability, and maintenance of a core vision in software-induced ecologies have become highly important. Requirements traceability within such evolutionary processes often involves runtime monitoring towards the requirements. Finally, new design goals enter often the stage. For example, industrial design ideals such as innovativeness or esthetics of the user experience play a larger role than just meeting sheer functionality.

**Capability-based evolution**: for the past 20 years, the dominant trend has been business process modeling and optimization that go together within monolithic COTS systems, or software product lines with relatively closed sets of requirements. When the nature of design requirements problem and the specification problem are fundamentally changing this concept of optimization and fixing designs will not do. In contrast, we need to understand how creation of capabilities drives and constraints requirements evolution. We are therefore witnessing a move towards evolutionary platforms, which are defined by complex consumption and production lattices (and related enabling architectures). Such networks need to self-organize or find ways to organize in flexible and loosely coupled manner as to define their core capabilities, and seek opportunities (often initiated from the edge) to exploit for market or process innovations in a speedy and flexible manner. Capability-based evolution defines core capabilities that can be competitively delivered with and by the software induced networks. These platforms also hold the networks together. By fixing core requirements and the related architecture for the efficient core implementation, the ecology designer makes bets on the speed and variability of change related to alternative requirements sets, which need to be delivered with an evolving set of capabilities (and the network). For example, software platform strategies have become a critical success factor in the automotive industry. However, this advantage can also turn into a deadly trap, when the company must be split in a manner orthogonal to the core original platform, assumed requirements evolution, and related networks. Ideas of evolutionary platforms extend thus beyond the notion of product lines. In addition to managing variability evolution, they need to consider distinctions between core processes (supported in the platform) and context processes (around it) and their dynamism. These distinctions must be based on a careful analysis of market power and network strength, and anticipate evolution in the underlying technological standards and architectures. However, as evidenced by the success of platforms such as Amazon, Google or Facebook, a well designed platform offers numerous opportunities for new types of requirements engineering based on system monitoring, e.g., social network analysis.

**Managing business semantics and new task variety**: due to *predictability problem*, the evolution of most systems must be aligned more closely with the change in the business process or system and its punctuations. Service oriented architectures, possibly combined with model-driven code and test generation, are now reasonably well established at the implementation level to do the job. The situation is quite different, however, at the level of business services, despite continued attempts for their standardization. The decomposition of monolithic business processes or product architectures (e.g., in cars) into more loosely configurable business services (or modules)
has turned out to be a far more complex task than anticipated due to the need to make semantics explicit that were hitherto hidden in the code, or people’s heads. This is, however, not only true for runtime service configurations, but also when outsourcing and especially when off-shoring. The design challenge is how to tackle the domain of (business) semantics, which is often culture dependent. Intercultural competencies often become necessary for requirements engineers operating in these new environments. Legal compliance is also forming an increasingly challenging aspect, not only in terms of how to protect my intellectual property (IP) embedded in code, or requirements (or whether even to protect it), but also in terms of protecting oneself against being sued due to imprecise contractual agreements or for example, not honoring open source licenses in some parts of code buried in the implementation.

To summarize, the good news of our foray into the world of RE is that RE has never been more important and its criticality will continue to grow. The bad news is that RE is a different endeavor now. Accordingly, we need to approach RE in new ways where we intertwine requirements and contexts, evolve designs and ecologies, manage through architectures and learn to recognize and mitigate against design complexity. Consequently, we need to expand RE research into new directions – including complexity science, industrial design, organization design, and behavioral economics – and engage these fields in an intellectual exchange on why and how design requirements matter in the design of complex software within a living – and therefore evolving – world.

Acknowledgments

We thank Barbara Pernici and three anonymous reviewers for constructive criticisms which lead to significant improvement in the papers argument and organization. We thank Sean Hansen, Nicholas Berente, Dominik Schmitz, and Anna Glukhova for helping to organize the workshops, and the workshop participants for inspiring discussions and Sol Greenspan for constructive comments on earlier drafts of the manuscript. This research was in part funded by National Science Foundation’s “Science of Design” initiative, Grant no. CCF0613606, by DFG projects CONTICI and the UMIC Excellence Cluster; also by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References


