# Requirements in the 21st Century: Current Practice and Emerging Trends*

Sean Hansen, Nicholas Berente, and Kalle Lyytinen

Case Western Reserve University, 10900 Euclid Avenue,
Cleveland, Ohio, USA
hansen@case.edu, berente@case.edu, kalle@case.edu

**Abstract.** Requirements have remained one of the grand challenges in the design of software intensive systems. In this paper we review the main strands of requirements research over the past two decades and identify persistent and new challenges. Based on a field study that involved interviews of over 30 leading IT professionals involved in large and complex software design and implementation initiatives, we review the current state-of-the-art in the practice of design requirements management. We observe significant progress in the deployment of modeling methods, tools, risk-driven design, and user involvement. We note nine emerging themes and challenges in the requirement management arena: 1) business process focus, 2) systems transparency, 3) integration focus, 4) distributed requirements, 5) layered requirements, 6) criticality of information architectures, 7) increased deployment of COTS and software components, 8) design fluidity and 9) interdependent complexity. Several research challenges and new avenues for research are noted in the discovery, specification, and validation of requirements in light of these requirements features.

**Keywords:** Requirements, modeling, specification, validation, verification, change, large systems, complexity, stakeholders, field study.

## 1 Introduction

The first step in any design effort is to ask what it is that one intends to create: What objectives does it need to address? What must it be capable of doing? Who will it serve and how? To what constraints must it conform? These questions are fundamental to design in its myriad forms – industrial design, graphic design, instructional design, and business process design, among others [1, 2]. As we know from past research and practice, software design is no different in this regard. In this paper, we refer to tasks in the design of software-intensive systems where questions of this nature are addressed as the management of design requirements.

Design requirements represent a crossroads where several research, business, engineering, and artistic communities converge. Therefore design requirements discussions span a range of research disciplines, including computer science, information

* This research was funded by NSF Research Grant No. CCF0613606. The opinions expressed are those of the researchers.

K. Lyytinen et al. (Eds.): Design Requirements Workshop, LNBIP 14, pp. 44–87, 2009.

systems, new product development, marketing, strategy, organizational theory, and a variety of engineering fields. In addition, a number of social science inquiries, including cognitive psychology, anthropology, sociology, and linguistics are relevant for the issues raised [3]. Not surprisingly, these diverse research communities do not always communicate well even when their core phenomenon of interest is largely shared. This diversity of research backgrounds is reflected in the rich variety of terms that have been employed to characterize the requirements arena. Requirements definition [4, 5], requirements analysis [6, 7], requirements determination [8, 9], requirements development [10, 11], requirements engineering [12, 13], and systems analysis [14, 15] have all been used to capture facets of the design requirements task. Outside software systems, the term requirements is often eschewed entirely in favor of *needs* or *customer attributes* [16]. For the purposes of the current study, we use the term *design requirements processes* to refer to the range of activities involved in determining what features and functions an artifact must embody and what constraints it must satisfy in order to address the types of questions outlined above. We will employ this term to emphasize the universal nature of requirements questions for contemporary software-intensive design efforts.

Despite the fact that design requirements form an interdisciplinary area of study [17, 18], the bulk of research on the subject comes from software engineering, computer science, and information systems domains. Within these communities, the criticality of requirements processes has been recognized for decades. In one of the earliest works to raise requirements questions, Ross & Schoman [4] stated that inadequate attention to the needs and envisioned functions of a system leads to "skyrocketing costs, missed schedules, waste and duplication, disgruntled users, and an endless serious of patches and repairs euphemistically call 'system maintenance'" (p. 6). A similar point was made by Bell & Thayer [19], who noted that problems originating in the requirements process often go undetected and later get attributed to bad design or technological limitations. The economic ramifications of requirements were recognized early on by Boehm [20] when he noted that the correction of requirements errors cost a fraction of the impact when errors go undetected until testing and implementation. Later, Boehm & Papaccio [21] mapped empirically the exponential rise in the cost of requirements errors as a systems development effort progressed.

Two decades ago, researchers had already highlighted many of the challenges associated with the requirements undertaking itself. Davis [8] observed that requirements challenges are inherent in any systems design effort because of the complexity of the requirements task, the limits to human information processing, and the intricate interaction between designers and intended users. The emergence of adversarial relationships between designers and other stakeholders has often been cited as a key impediment to effective requirements processes [22]. Even when good relationships have been established, the requirements processes are often inhibited because users do not thoroughly understand what they want to achieve [23]. In addition, the process remains sensitive to other forces that shape organizational life. Bergman et al. [24] noted that requirements processes are unavoidably intertwined with the politics of resource allocation and legitimacy of decision-making within organizational environments.

Ultimately, design requirements processes are challenging due to their Janus-faced nature.[1] Throughout the requirements effort, designers direct their gaze simultaneously in two opposite directions and toward two different social worlds: backwards to the needs of the stakeholders for whom they are designing an artifact, and forwards to the artifact itself and the demands set up by the development environment. Design requirements represent the gate or trading zone in the process at which the amorphous and ambiguous needs of a business or a consumer are married with the concrete design and engineering steps needed to address them [3, 18].

Despite a significant body of research on requirements, unresolved issues continue to haunt designers across the industrial spectrum. In particular, the "requirements mess" remains a challenge among information technology professionals [25]. Since the Standish Group first published its survey of information systems success and (more notably) failure [26], requirements researchers have been quick to note that the three leading sources of project difficulty – i.e., lack of user input, incomplete requirements, and changing specifications – are directly related to the creation and management of a projects' design requirements [11, 17, 27-29]. Likewise, Keil, et al. [30] observed that misunderstanding of requirements and the failure to gain user involvement were among the top project risks. In addition, researchers have noted the persistent gap between research and practice, despite the fact that the area of inquiry is ostensibly motivated by the real-world concerns of designers [3, 31-35]. This gap runs both ways: practitioners are slow to adopt the requirements methods developed by researchers [36], whereas researchers often turn a blind eye to the actual practices and needs of designers [37].

The present study seeks to address this discontinuity through a review of major threads in the past research into design requirements. We strive to assess the state-of-the-art in requirements practice and theory, identify gaps between research and practice, and solicit fruitful avenues for research in the coming years. The research questions that we seek to answer are diverse:

1. What activities and assumptions characterize the contemporary practices of managing design requirements?
2. How are requirements practices consistent with perspectives on design requirements, as reflected in the research literature?
3. What tasks are typical in current requirements processes and what are the newly emerging challenges?
4. What trends are driving requirements practice changes today and over the coming years?

To address these questions we report the findings of a field study about requirements practices among leading design professionals from across the industrial spectrum. We seek to glean key insights about the state of current practice and identify drivers of change in 21st century requirements design efforts.

The remainder of the study is organized as follows. In Section 2, we review the research literature, and introduce central concepts and topics that will inform our study.

[1] Janus was the Roman god of gateways, doorways, beginnings, and ends. This is a fitting metaphor for requirements researchers, who stand now at the threshold of a new era in requirements practice.

Section 3 explains the research approach adopted and research questions that we sought to address. Section 4 highlights key findings from the field study. The implications of these findings for the future of design requirements research is offered in Section 5. Section 6 concludes the study with a call to action for researchers and practitioners alike.

## 2 Requirements Research – A Short Overview

Before exploring the state-of-the-art in requirements practice, it is essential to understand the discourse that has emerged around requirements within the research literature. Accordingly, we will attempt to highlight some of the key concepts that have marked the requirements research tradition. As noted above, requirements processes have been implicated in a wide variety of design shortcomings. As a result, the research around requirements has remained predominantly prescriptive. It is replete with analytical frameworks, standards for requirements quality, elicitation approaches, and modeling methodologies. A wide array of textbooks and reviews have been published, advising practitioners on the most advisable approaches to requirements engineering [10, 12, 38-43]. By comparison, a relatively small percentage of the literature has focused on advancing a theoretical or empirical understanding of how design requirements are discovered, defined, negotiated, and managed by individuals and teams within organizations and why these processes are so difficult. Moreover, the prescriptive modeling and process methodologies have seldom been subjected to rigorous empirical scrutiny due to issues of cost, access, and threats to internal validity [44].

However, it is important to note that requirements processes are far from monolithic. Just as requirements represent one facet of a broader design effort, so too requirements processes can be divided into a number of facets. Within the research literature, multiple frameworks have been developed, positing anywhere from two to seven primary facets for requirements [45]. For the current discussion, we adopt a widely-employed and straightforward categorization of the requirements processes into three facets: 1) discovery, 2) specification, and 3) validation & verification (adapted from [39]).

During *discovery*, designers develop an understanding of the application domain and infer specific design needs through consultation with stakeholders and reviews of other sources of information [12]. This process includes the identification of all relevant stakeholders for the design effort. *Requirements specification* is a term that is treated both as a noun and a verb within the research literature. As a noun, a specification forms the document in which the requirements for a design effort are articulated, and it represents the fundamental agreement between the stakeholders and the design team [41, 46]. The verb form suggests the process of developing and managing the specification document; it is the process by which the design team abstracts and represents the requirements for the design effort [39, 44]. This interpretation of requirements specification as a process will be primarily used in the current discussion. Finally, during *requirements validation and verification* designers ensure that the requirements are of high quality, address the users' needs, are appropriate for the design effort, and have no inconsistencies or errors [47].

While this tripartite characterization appears to imply a linear approach, the three facets are normally employed iteratively, often moving progressively to more detailed levels [45]. The degree of iteration between the facets varies based on the methodology espoused by the design team. However, despite the strong interconnectedness of facets, most requirements research has focused on only one of them at a time. A more detailed exploration of these facets is warranted. Next we will highlight ideas that have emerged in each of these facets, acknowledge assumptions associated with each, and discuss persistent challenges to be explored.

### 2.1 Discovery

Discovery is the first component of any design effort – a designer or a design team must determine what organizational or customer needs must be addressed by the design artifact [13, 39, 48]. This process is also often referred to as requirements elicitation which conveys a widely held (i.e., traditional) position that knowledge about requirements resides with users or other stakeholders, and must be "teased" out and clearly articulated by the designer.[2] Discovery is also the primary process by which designers gain knowledge of the relevant application domain. As Loucopoulos and Karakostas [39] note, the critical role of 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 statement illustrates the assumption that the designer is in most cases regarded as an outside party in the application domain, who is brought in for a limited period of time to resolve a problem that is of potential concern to others.

While one may speak of several traditional approaches to discovery, there are a wide range of techniques that have been employed in this undertaking [32, 48]. Table 1 summarizes a number of key discovery techniques and their associated advantages and disadvantages. The most rudimentary form of requirements discovery is introspection on the part of designers [48]. During introspection, designers reflect upon or imagine design features that they would find desirable given their understanding of the application domain. Such introspection does not involve direct discussion with other design stakeholders and is therefore often discouraged, if divorced from interactive techniques. Among the most widely noted discovery techniques are one-on-one interviews between a designer and stakeholder, focus group discussions facilitated by members of the design team, and direct observation of business processes or stakeholder activities [32, 49]. Interviews and focus groups emphasize a discussion between representatives of the design team and those closest to the application domain around current experience, areas of discontent with the existing environment, and desired changes that a design artifact might engender. These methods involve both a scrutiny of the current state and generation of possible future states that could be pursued during the design undertaking. Direct observation eliminates explicit discussions, but underscores a designer's detailed understanding of the ways in which activities actually unfold in practice.

[2] The term *discovery* was adopted in an effort to broaden the understanding of requirements identification to cover envisioning or innovation on the part of design team members. This conception is meant to overcome the limitations of the passive "collection" or "capture" role reflected in the phrase *requirements elicitation*.

A number of data-intensive discovery techniques, such as protocol analysis [50, 51] and the use of ethnography [48, 52, 53], have been proposed to enhance identification and assimilation of tacit information during requirements processes. Finally, prototyping has been widely employed as a way to expand requirements elicitation activities. It refers to the development of a rough and relatively rudimentary design artifact that includes some essential features desired by relevant stakeholders [54]. Prototyping is particularly effective in establishing a common basis for understanding and communicating design ideas between a designer and stakeholders. For this reason, it may also be analyzed within the requirements validation facet.

While a wide array of discovery techniques are available, it is important to note that they are not mutually exclusive and a combination of techniques can be complementary [17, 32]. It has repeatedly been observed that no single technique is appropriate for all design contexts [13, 29, 55, 56]. There is also clear empirical evidence that the way in which the discovery process is structured impacts both the quality and quantity of the requirements, as a combination of techniques enable designers to adopt multiple perspectives on the application domain [57]. In addition, Hickey & Davis [29] note that the careful selection of appropriate techniques for a given situation is the hallmark of a truly experienced analyst. Regardless of the methods adopted, the process should be well aligned with the documentation of those requirements.

Despite the proliferation of requirements discovery techniques, several questions remain to be answered. It is unclear the degree to which espoused discovery practices have been adopted in real-world design efforts and under what conditions. As with many areas of social science research, requirements discovery is marked by a significant gap between research and practice [32, 33]. There is some evidence that formal discovery techniques have been effectively applied by technology consultants and expert users [29], but their degree of acceptance in a broader industrial context remains an open question. Other areas ripe for inquiry include: What skills do design team members need to effectively execute various discovery techniques? In the area of software engineering, what impact has the rise of commercial off-the-shelf (COTS) solutions had on approaches to requirements discovery within organizations? Do most designers adopt a one-shot approach or a more incremental perspective on requirements discovery? How has the need for speed and agility altered requirements discovery?

### 2.2 Specification

As stakeholders needs emerge, they must be rendered in some concrete format and representational scheme. This rendering effort is referred to as the specification process. Overall, a requirements specification supports interpretation and understanding among all design stakeholders around what the artifact is supposed to accomplish, while at the same time laying a sufficient technical foundation for the subsequent development effort. Thus, specification is more than just rendering requirements into some standardized format from the information expressed by stakeholders. It marks the point of transition where the stated needs of stakeholders will be extended with the functional and technical implications that flow from them. Nowhere is the Janus-faced nature of design requirements more evident than in the specification. Traditionally, the requirements literature has sought to emphasize the backward focus towards

**Table 1.** Summary of Selected Requirements Discovery Techniques

| *Discovery Techniques* | *Summary* | *Advantages* | *Limitations* |
|---|---|---|---|
| Designer Introspection | Designers' reflect or imagine features that they would find desirable given their understanding of the application domain | Requires no specialized elicitation skills on the part of design team members<br>Essential in innovative designs which break out from established approaches | Eliminates contact with other design stakeholders<br>Ignores the needs of those most closely linked to an application domain<br>Provides no basis for validation of requirements |
| Interviewing | One-on-one discussions between a user and designer using open-ended/ unstructured, semi-structured, structured, and survey-based variants [32, 48, 49] | Effective for gathering large amounts of information about the application domain [32] Enables designers to focus on a limited number of users as representatives of other stakeholders<br>Requires fewer specialized skills than other discovery techniques | Stakeholders are constrained by the line of questioning employed by the designer [48]<br>Biases in questioning and anchoring effects direct the inquiry to the preferences of designers rather than the needs of stakeholders [58, 59]<br>Gets only at work practices that can be explicitly expressed [8, 60]<br>Appropriateness of the sampling and access to stakeholders are critical |
| Focus Groups | Designer-facilitated inquiry with a selected group of stakeholders about the current state of practice and the future design space;<br>Adapted from marketing research [61] | By moving away from the individual focus groups engender a more thorough exploration; a statement by one participant may prompt conflicts, extensions and responses by others<br>The presence of multiple stakeholders allows for the questioning and exploration of the assumptions and timely attention to areas of conflict | Designer/analyst facilitation may limit the conversation to the topics determined a priori by the design team<br>Stakeholders are called upon to reflect in abstract on their practices and tacit features of the context remain unexplored<br>Due to a representation from multiple stakeholder, the potential for destructive conflict and political maneuvering is raised<br>Appropriateness of sample is still a concern and the perceived costs to the organization are often higher |

**Table 2.** Summary of Selected Requirements Discovery Techniques (*continued*)

| *Discovery Techniques* | *Summary* | *Advantages* | *Limitations* |
|---|---|---|---|
| Protocol Analysis | A stakeholder is asked to perform an activity and talk aloud about the steps – outlining the rationale for each action [50];<br>Grew often out of the development of expert systems [51] | Can augment an interview process by surfacing tacit elements of work<br>Engenders a more reflective and thorough description on the part of the stakeholder. | Is built upon an overly-simplistic, computational model of cognitive processes,<br>Apt to overlook nuances of activity in an actual context of use [48]. |
| Prototyping | The development of an early, rudimentary version of system that includes the essential features [54]. | Assists when requirements are poorly understood by enabling stakeholders to get experience of what a new artifact may be like<br>Promotes discussion of system features that had not been considered during interviewing or group discussions [12].<br>Creates a common point of reference [54]. | Users become attached to functionality provided in a prototype and may resist changes to the proposed design [54]<br>By emphasizing iteration prototyping may result in "spaghetti code," [62, 63].<br>Problematic in the development of large systems having significant interdependencies with other systems [54]. |
| Ethnographic Methods | Longitudinal observation within the application domain;<br>Adapted from ethnomethodology in sociology and anthropology [64, 65], and inspired by advances in industrial design [2] | Ethnographic methods can discover knowledge of the application domain to a degree not achieved with traditional methods [48, 52]<br>Mitigates the difficulties associated with tacit knowledge because designers experience the application domain not | Consumes significant time and resources because of long-term focus<br>May be deemed infeasible for design efforts with short timelines or tight cost restrictions |

the needs of stakeholders by stating that requirements are concerned with what is to be achieved by a design artifact (i.e., the "what") without regard to the manner in which it will be designed and implemented (i.e., the "how") [38]. Yet this stance "leaves unresolved the question of whether or not it is possible or desirable to separate the 'what' from the 'how' in practice" [66: 18]. With rising systems complexity and interdependence between systems, scholars have started to acknowledge the need for incorporating design considerations and key constraints on the design space during specification [39, 67].

Before discussing in more detail the primary treatments of specifications in the extant research literature, it is worthwhile to introduce a number of concepts that are central to the discussion of requirements specifications:

*Abstraction* refers to the ability to glean the essence of something from specific instances [45]. In the context of design requirements processes, abstraction enables designers to induce essential elements or processes from specific statements about the application domain and problem space. This helps to ensure that information which enters the specification is essential rather than idiosyncratic, and offers a sound baseline for design.

*Decomposition* is the process by which systems are partitioned into components. It is a critical capability in any complex design because it allows members of a design team to focus their efforts on manageable tasks. In addition it breaks a large design into its composite subsystems and supports designer's ability to explain and predict outcomes. Decomposition lies at the heart of contemporary advances in modular design and economies of scale and scope in design [68].

*Traceability* refers to the idea that all "lower" level statements of requirements should be associated with specific higher order objectives and properties and vice versa [69, 70]. In effect, there are two forms of traceability, which correspond to the two directions of the Janus's gaze. *Backward traceability* is the ability to tie a stated requirement and its design and implementation back to its source in business objectives. *Forward traceability* refers to the ability to trace a given requirement or feature to the components of the designed artifact or their interactions that ultimately address it [71]. The traceability concept is the compliment of decomposition. In design, traceability is essential to manage complexity and change and to guarantee that systems validate and "meet" requirements. It also enables designers to evaluate the implications of requirements change regardless of the level of detail at which they are introduced. Finally, traceability facilitates the assessment of completeness and consistency of requirements (see Validation & Verification).

In the development of a requirements specification document, designers generally combine natural language descriptions with formal or semi-formal models of the application, problem, or design space.

**Natural Language.** During discovery, the primary way in which stakeholders express their needs is through natural language. Accordingly, design requirements at the highest level (i.e., business or user requirements) are rendered through natural language descriptions. Natural language use has several distinct benefits. Foremost among these is that most stakeholders prefer natural language to more formal specifications [72]. Natural language also provides a common basis for communications between the stakeholders and designers (as well between different

stakeholders), and it can provide a great deal of information about the contexts of use [73, 74]. Finally, natural language use is inevitable as we can never achieve fully formalized articulations [73]. Natural language remains the ultimate meta-language where all meanings must be negotiated, and it thereby offers openness to sense-making and discovery [75].

Despite its strengths, most discussions of natural language in requirements research have emphasized the challenges it presents and have proposed ways to overcome its limitations through formal analysis. Researchers have long argued that the informal treatment of specifications leads to ambiguity, incompleteness, and inaccuracy [76]. Ambiguity arises because stakeholders and designers may interpret the same words in different ways. Similarly, distinct stakeholders may use the same term differently, leaving designers to decide which sense is appropriate for the design context. Questions regarding completeness and accuracy emerge because the informal nature of natural language inhibits explicit analysis. Finally, natural language descriptions hide inconsistencies because they provide little basis for direct comparison across statements. In an effort to overcome such shortcomings, researchers have pursued natural language processing capabilities to automate the generation of formal models from natural language inputs [77-79]. However, the bulk of the specifications research has focused on ways to augment natural language representations with formal and semi-formal models of requirements.[3]

**Modeling.** Perhaps no single subject within requirements research has received more attention than that of modeling [17]. Some even argue that model development lies at the very core of the entire requirements undertaking [80]. In this context, modeling refers to the creation of abstracted representations (i.e., models) of the real world through the use of limited and established symbol systems [81]. The portion of the real world to be modeled is the application domain and its relationships with the proposed design. The resulting models reflect abstractions, assumptions, and known constraints within that design domain [39].

There are several key benefits that have been attributed to formal specifications. By encapsulating large amounts of information, requirements models establish a baseline of understanding. In addition, they may facilitate communication between distinct stakeholder groups [80]. Models also enable formal analysis to identify unstated requirements, predict behavior, determine inconsistencies between requirements, and check for accuracy. Finally, models serve to simplify the application domain by focusing on essential features in line with the principles of abstraction and decomposition. While each of the proposed benefits of modeling is sound in itself, these arguments illustrate one of the tacit assumptions that plagues much of the modeling literature – an emphasis on the perspective of the designer. Within this literature, the focus is squarely placed on the ways in which modeling can be used to support or enhance the work of designers with less regard for the preferences of other stakeholders.

Models are developed at multiple levels of detail. Loucopoulos and Karakostas [39] identify three central levels of modeling in contemporary design efforts: enterprise modeling, functional requirements modeling, and non-functional requirements

[3] There has been markedly less discussion about the converse potential of overcoming the limitations of formal modeling techniques through innovative uses of natural language.

modeling. *Enterprise modeling* refers to the development of models to reflect the broader organizational or market context of a design, including the representation of relevant stakeholder groups and the social structure, critical processes, and guiding objectives of the enterprise or the marketplace. Enterprise model development helps

**Table 3.** Summary of Modeling Meta Models

| Meta-Model Category | Description | Exemplars |
|---|---|---|
| State Models | Modeling a system as a set of distinct states and the modes of transition between states; Appropriate for representing reactive, event-driven systems. | ▪ Finite state machines [90]<br>▪ Petri nets [91]<br>▪ Statecharts [92] |
| Structural Models | Modeling of a system based on the structural features of the application domain; One of the earliest efforts at formal systems modeling, | ▪ Structured analysis and design techniques (SADT; [4, 93] |
| Activity Models | Modeling a system as a collection of activities; Appropriate for modeling "systems where data are affected by a sequence of transformations at a constant rate" (Machado et al. [88],p. 25). | ▪ Structured analysis and structured design (SASD; [94, 95] tools such as data flow diagrams (DFD) |
| Object-Oriented Models | Approaches that incorporate many concepts and fundamental techniques introduced in other methods; Adds to these concepts such as decomposition into objects, inheritance, and encapsulation [96]. | ▪ Object modeling technique (OMT; [97]<br>▪ Object-oriented software engineering (OOSE; [98]<br>▪ Unified Modeling Language (UML) [84] |
| Agent-Based Models | Modeling of complex systems as a collection of autonomous decision-making agents; Especially useful for the simulation of emergent phenomena [99] | ▪ Axelrod Cultural Model (ACM) [100]<br>▪ Construct-TM [101, 102]<br>▪ Sugarscape [103] |
| Goal-Oriented Models | Modeling of the underlying objectives that motivate a design effort; Goal-oriented models incorporate both goal features and linkages between distinct goals [104] | ▪ KAOS methodology [105, 106]<br>▪ Non-Functional Requirements (NFR) framework [89, 107] |

achieve a thorough understanding of the application domain and the interdependencies that it embodies. In the context of information systems development, enterprise models focus on interactions between a system and its environment. Examples of these types of models include rich pictures [82, 83], use cases [10, 84], business process modeling [85], and enterprise-level architectural modeling [86].

*Functional requirements modeling* focuses explicitly on representing requirements about the design artifact itself – abstracting it from the environment and making it amenable to design. Techniques for modeling functional design requirements have proliferated since the earliest efforts in the mid-1970s. Most of these modeling approaches are specific to the context of information systems design. The modeling techniques may be categorized based on the ontological perspectives they apply to the application domain [87]. Machado, et al. [88] refer to these ontological categories as meta-model characterizations. Table 2 provides a summary of meta-model categories and some of the associated modeling approaches. Finally, *non-functional requirements modeling* refers to the development of models to identify the constraints or restrictions on the design domain. In the information systems development discourse, non-functional requirements also incorporate the quality expectations for a system, often referred to collectively as "ilities" (e.g., reliability, adaptability; [89].

The bulk of the modeling literature has focused on techniques for modeling functional design requirements. While most of these modeling methods were introduced as individual techniques for representing an application domain, recent trends have been toward integrating across modeling perspectives [39, 108]. For example, the IDEF family of modeling methods enables a design team to apply multiple development ontologies to the requirements modeling [109]. The introduction of the Unified Modeling Language (UML) during the last decade has greatly extended the trend towards employing multiple perspectives. UML is an outgrowth of the object-oriented specification tradition, but incorporates a broad suite of modeling techniques, including class diagrams (an extension of E-R diagrams), state-chart diagrams, activity diagrams, and use case diagrams [84, 110].

In addition to the move toward integration across ontological perspectives, modeling research has been marked by two countervailing trends. The first emphasizes increased standardization in the specification of notation systems and processes. Hundreds of modeling techniques have emerged over the past 30 years, but this diversity in fact poses an impediment to adoption. Some researchers have called for a moratorium on new model development until existing models have been tried and exhausted by practitioners [36]. The development and adoption of UML provides an example of the benefits of standardization. The UML suite was developed when three "thought leaders" in object-oriented modeling recognized the convergence in their modeling methods and decided to work together to create an industry standard [84]. Since its introduction, UML has rapidly emerged as a de facto industry standard, creating a measure of modeling consistency across industries, organizations, and design environments backed by standardization organizations [111].

The second, and perhaps contradictory, trend is the move toward increased customization of modeling based on the types of systems or contexts involved. *Situational method engineering* (SEM) is a movement to customize development and modeling approaches to the needs of a given design task through the selection, recombination, reconfiguration, and adaptation of *method fragments*, many of which are

modularized abstractions of existing methods [112, 113]. Similarly, recent work on domain-specific modeling (DSM) emphasizes efficiencies to be gained by tailoring languages and modeling techniques to the specific vocabularies and abstractions of a given domain [114, 115]. While the trend toward customization focuses mainly on the composition and reconfiguration of methods, it has significant implications also for the evolution of requirements modeling tools and techniques [116]. The observation of these two trends raises the question – Can increased standardization be reconciled with desires for customization in modeling techniques and how do such goals align with specific needs in the future?

The conflict between these trends is but one of the pressing questions in research around requirements specification. Other important issues include the following: With significant adoption of UML is there still a need for novel approaches to the modeling of design requirements? Which components of the UML suite or other techniques have been adopted by design practitioners and why? Has the adoption of formal modeling techniques engendered a substantive improvement in requirements and design quality? How can different models be practically integrated? Turning again the issue of natural language, little attention has yet been paid to ways in which language and communication skills of design professionals could or should be enhanced to support high-quality requirements specification. These are among the issues that must be addressed by research on requirements specification in the coming years.

### 2.3 Validation and Verification

*Validation and verification* addresses the question of whether or not the requirements processes have been conducted effectively and the degree to which the specifications will support a productive design effort. Some researchers use only the term 'validation' or 'verification' when discussing this facet, but an important nuance between these two sides prevail. *Validation* is the effort to ensure that requirements accurately reflect the intentions of the stakeholders [117]. *Verification* focuses on the degree to which requirements conform to accepted standards of requirements quality [10, 47]. Boehm [47] captures the distinction succinctly when he states that validation addresses the question "Am I building the right product?"; while verification asks "Am I building the product right?" (p. 75).

**Validation.** Locating requirements validation as the end point of the design requirements may give a false impression that it is of limited importance and does not shape behaviors significantly. In truth, the validation begins almost simultaneously with discovery and continues through the specification. When a designer uses paraphrasing to check his or her understanding of a stakeholder's request or statement, validation is taking place. Indeed, one of the primary approaches to requirements discovery – namely prototyping – is often referred to as a key validation technique [39] .

One of the central issues in requirements validation is the potential for disagreement between individuals or stakeholders groups. Given diversity in their backgrounds, roles, and values, it should not be surprising that conflicts frequently emerge [17, 118, 119]. Effective management and resolution of such conflicts is essential if the design effort is to advance. A range of techniques have been proposed to help designers with conflict resolution:

*Requirements prioritization* refers to the process of determining the relative value of individual or sets of design requirements [120]. By assigning values to requirements, designers establish a mechanism for mediating between requirements conflicts that arise. Berander and Andrews [120] identify a number of prioritization techniques that have been applied, including numerical assignment (i.e., priority grouping), analytical hierarchy process [121], cumulative voting, and stack ranking.

*Requirements negotiation* involves the identification and resolution of conflict through exploration of the range of possibilities available [119, 122, 123].[4] These negotiations often draw upon fields of research on multiple criteria decision making and game theory [123, 125], and apply group support systems or collaborative environments for effective negotiation around requirements conflicts [118, 126-128].

*Viewpoint Resolution* builds upon the *viewpoints* thread within requirements research. *Viewpoints* refer to the emphasis on obtaining design requirements from individuals and groups having different perspectives on the design [129]. Leite and Freeman [130] introduce *viewpoints resolution* as a "a process which identifies discrepancies between two different viewpoints, classifies and evaluates those discrepancies, and integrates the alternative solutions into a single representation" (p. 1255). Thereby, viewpoint resolution feeds back into the modeling process by developing a model of requirements conflict.

**Verification.** The counterpart to validation is verification. With verification, we turn our attention back to the functional and technical implications of the requirements. Much of the discussion around requirements verification focuses on ensuring adherence to standards of requirements quality, including consistency, feasibility, traceability, and the absence of ambiguity [10]. *Consistency* refers to the idea that requirements should not conflict with the overall objectives of the design effort, or with each other. As the number of individual requirements statements proliferate in large scale projects, concerns over the potential for inconsistencies between statements rise and verification measures are implemented in efforts to safeguard against errors. *Feasibility* is the degree to which a given requirement can be satisfactorily addressed within the design environment of an organization. This includes not only a consideration of whether or not an artifact can be developed in line with the requirement, but also how it can be subsequently maintained. As discussed above, *traceability* is the degree to which individual requirements can be tied to both higher order objectives and detailed elements and their interactions within an artifact.

A number of techniques have been proposed to support this verification function. In one of the earliest discussions of the importance of verification (and validation), Boehm [47] presented a range of both manual and automated approaches to verification, including such simple techniques as manual cross-referencing, the use of checklists, and scenario development to the more complex activities of mathematical proofs, detailed automated models, and prototype development. Other key verification techniques include formal inspections [131], structured walkthroughs [132], and automated consistency checking [133].

[4] It is worthwhile to note that many researchers would position requirements negotiations as part of the specification phase of a requirements process or as a distinct phase altogether (e.g., [124]).

In total, the requirements research literature has provided a wide range of insights into the individual facets of requirements work. From the introduction of formal modeling techniques to the development of novel approaches to discovery, requirements scholars have consistently sought to improve the lives and resources of designers. Yet, the degree to which these efforts have resonated with practicing designers remains to be seen. While some researchers emphasize the increasing adoption of techniques from research (e.g., [34]), others bemoan the growing gulf between researchers and the practicing design community (e.g., [33]). In addition, our review illustrates a number of key research assumptions including a focus on individual systems, attention to a single facet of the process at a time, emphasis on notations to represent requirements, and the primacy of a designer-centric perspective. In the next section, we analyze these assumptions in the context of a field study.

## 3 Research Approach

In an effort to explore the current state of requirements practices across a variety of organizational and industrial contexts, we conducted a series of semi-structured interviews with IT and design practitioners from the United States and Europe. The data collection efforts were structured around an interview protocol that was jointly developed by the researchers. The interview protocol was designed to elicit responses to a number of distinct aspects of the professionals' design experiences, including a discussion of current design requirements processes; perceived impediments to the identification, specification, and management of design requirements; drivers of change in requirements practices over the preceding five-year period; key trends within the market or technological environments relevant to the given organization; and envisioned changes to the practice of requirements in the near future. The core protocol remained constant throughout the data collection process, however, in line with the grounded theory concept of constant comparison, some questions were added to the protocol based on insights from the initial interviews [134]. In addition, interview participants were encouraged to express their thoughts on any topics which they felt were relevant to requirements processes and contemporary design environments.

To foster external validity and to address threats to the internal validity of the study, the research team sought participation from individuals and firms engaged in a wide variety of design environments. A number of business and design contexts were initially targeted in the site selection process to ensure representation from areas where the researchers expected to observe significant market and technological change occurring. To ensure representation from leading edge and mainstream organizations the research team sought participation from senior technology leaders within a range of Fortune 500 organizations. A total of 30 interviews were conducted with 39 individuals participating. The interviews included firms from a wide range of industries, including software and information technology, automotive manufacturing, industrial design, aerospace, telecommunications, professional services, and healthcare sectors. Importantly, despite this range of industry domains, all of the professionals interviewed are engaged in the design of software-intensive systems. In order to protect the confidentiality of the respondents, none of the quotes or statements from the interviews are attributed to specific individuals or firms.

These initial focal contexts of 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.

It should be noted that our sampling approach reflects a 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 data. Interview transcripts were coded using Atlas.ti, a qualitative analysis application. The interview protocol served as the preliminary coding structure for the data. However, in line with a grounded theory approach, additional codes were created as specific themes or recurring issues began to surface in the coding process [134]. The code structure was iteratively revised until the researchers determined that all relevant themes or issues were reflected [135]. Several of the interview transcripts were coded repeatedly as the final coding structure emerged. The aim of this analysis was to identify distinct patterns in current requirements processes as well as to observe emerging key themes and issues in the day-to-day practice of requirements work. In the Findings section we will explore these observations in detail.

# 4 Findings

## 4.1 Current Practice

The field study revealed a number of key observations regarding the current practice of requirements management. Several of these findings reflect general approaches to requirements determination issues while others relate to specific facets in the requirements process (e.g., discovery, specification, validation). We will briefly discuss the findings regarding current practices before delving into the emerging themes and issues that surfaced in the study. Table 3 provides a summary of our findings regarding current requirements practices.

**Table 4.** Current Requirements Practice

| | |
|---|---|
| **Overarching Practices** | Development based on the use of CASE tools |
| | Risk mitigation common |
| | Broad external and internals stakeholder involvement in requirements processes |
| | Focus on data and process modeling |
| | Non-distinction of requirements tasks & functional/nonfunctional requirements |
| | Contingent application of requirements methodologies |
| | Limited focus on stakeholder conflict |
| **Discovery Practices** | Primarily focus groups and interviews |
| | Simultaneous elicitation and validation |
| | Responsibility for requirements discovery and justification rests largely with business |
| **Specification Practices** | Specifications based on CASE tools |
| | Widespread application of use cases |
| | Natural language, data, and process modeling representations based on UML standard |
| **Validation & Verification Practices** | Little use of formal verification practices |
| | Widespread use of prototypes and group walkthrough sessions |
| | Common expectation of a formal stakeholder signoff |

### 4.2 Common Requirements Practice

Overall, requirements practices have evolved significantly over the past decade, often in line with prescriptions offered in the academic and consulting literature. For example, much of the requirements literature of the 1980s and 1990s prescribes a disciplined process, often emphasizing the control of development risks [8, 136]. In addition, there has been a significant emphasis on fostering the involvement of a variety of stakeholders and the application of formal modeling techniques such as UML, and the use of supporting CASE tools [137, 138]. Our data generally suggest practices which are consistent with most of these prescriptions. Development environments based on tools such as IBM's Rational Suite are commonplace. Several participants note that project risk mitigation is a central area of development focus, and some informants indicated that portfolio risks are consistently measured and actively managed. The individuals and teams interviewed indicated that requirements activities commonly include focus group discussions, cross-disciplinary project teams, and requirements sign-offs from an array of stakeholder groups. In addition to the widespread use of data models, several organizations note sophisticated process modeling activity, including the widespread application of use cases, even in situations where other elements of UML were not fully adopted. Other principles that are addressed in

the literature, such as traceability and structured process improvement (e.g., CMMI), while not prevalent in current design practices, received significant consideration in the future efforts of many interviewees, and were noted as directions in which firms are actively moving. Similarly, trends that are often addressed in both the academic and practitioner literature, such as web services/service-oriented architectures (SOA) and outsourcing of development, were reported as influencing current requirements practice.

Yet, in many cases designers did not employ concepts and distinctions that are common place in the research literature. For example, few of interviewees made a distinction between functional and non-functional requirements. While they do seek to capture constraints about desired performance (e.g., traditional "-ilities") for designs, they do not document and manage these requirements in a differential manner. Another break with the research is in the characterization of the requirements process. Several interviewees expressed their belief that requirements processes are indistinguishable from the design. While researchers have long asserted that requirements should presage efforts to determine how desired functionality could be achieved (i.e., the formal design), many participants felt that requirements questions are properly interspersed in the design process. Those interviewed emphasized the intensely iterative nature of requirements and design.[5] Even when the recognition of requirements as a formal, early phase of a design task was recognized, the designers did not mark distinctions in requirements activities, such as elicitation, specification, negotiation, validation, or verification. Thus, many of the classification schemes that demarcate discourses within requirements research remain unrecognized in contemporary practice.

A second key finding with respect to the practice of requirements is that the application of standardized and more formal methodologies might best be described as haphazard. Within the organizations represented, design professionals are seldom expected to adhere to explicit methodologies in the discovery, specification, or verification of design requirements.[6] Most projects advance through a patchwork of techniques at the discretion of project managers. Despite the generally idiosyncratic nature of the requirements processes, there were a number of contingencies for the application of methods. Project budgets, personnel needs, and the number of systems impacted are key considerations in determining whether or not a more rigid process is necessary, with the larger and integration-intensive efforts carrying the increased expectation of the use of formal methods. Interestingly, the application of formal methods throughout the design process and across different projects was repeatedly noted as a direction for future development and improvement. The following statement is characteristic:

---

[5] Note that while this iteration was often discussed, it was rarely in the context of agile development. Interviews were virtually devoid of discussions of agile methodologies, with only a couple of exceptions - a finding which may be a function of the highly complex development practices within our sample.

[6] It should be noted that validation efforts are an exception to this finding as formal mechanisms for phased advancement of design projects (in the form of sign-offs by business sponsors or other key stakeholders) were nearly universal.

> "You have to become a lot more method [sic], a lot more rigor, a lot more science and gathering your requirements, qualifying them and then quantifying them in terms of financial [considerations]. Because at the end of the day, that's what we're in business for, to make money and pay dividends to our shareholders."

Another consistent finding from the study pertains to the management of conflict within designs. Despite the fact that researchers pay significant attention to negotiation and the management of disputes, conflict between stakeholders was viewed as largely unproblematic. Simple prioritization of requirements by a key sponsor or stakeholder acts as a primary mechanism for the management of conflicts. Frequently, the determination of such priorities is tied directly to the funding – i.e., whoever is perceived to be the primary source of funding sets the priorities. In this way, the valuation of requirements is transferred from the design team to the business stakeholders. However, the voice of IT stakeholders remains significant when the prioritization is subject to prior architectural decisions and constraints (see "Key Themes and Issues" section).

The participants experienced the most significant impediments to effective requirements processes in the interpersonal aspects of a design effort. In large part, these challenges reflect those often noted as key challenges throughout the requirements literature: stakeholders not knowing what they want, the inability of stakeholders to articulate what they want even when they do know it, and limitations in the communication skills of the design team. Interestingly, respondents noted very few impediments arising from available technical resources and formal methods. For example, no single participant felt that the absence of appropriate modeling approaches and tools set up a significant challenge to their requirements processes.

The study discovered a number of key findings concerning specific facets of the requirements processes. While the respondents themselves frequently failed to distinguish such requirements activities as discovery, specification, modeling, verification, and validation, the interview protocol helped glean insights regarding their approaches to the dimensions recognized in the protocol. By applying this established lens, we are able to discern linkages and discontinuities between current practice and past requirements research.

**Discovery.** With regard to discovery techniques, one of the most consistent observations regarding the process by which design teams explore and understand the needs of stakeholders is the relatively narrow range of techniques employed. Most organizations relied only on focus groups and other group-based discussions as a primary mechanism for requirements discovery. One-on-one interviews with stakeholders are also common. Although more intensive measures such as protocol analysis, direct observation of work practice, and ethnographic participation in the application domain were noted by a small number of respondents, traditional discursive techniques continue to dominate.[7] Also, we noted that discovery and validation often occurred simultaneously – frequently in the form of prototyping but also in other forms such as "blueprinting" sessions.

[7] It is worth noting that firms adopting less traditional discovery approaches are specifically recognized within design communities for their unorthodox perspective on requirements gathering.

A second key finding is the degree to which requirements articulation has been established as the responsibility of the relevant line-of-business or other stakeholder group. In several firms, sponsoring stakeholders are expected to engage the design team with a thorough statement of needs, extending beyond the business requirements to high-level functional requirements. In one case, a firm had started a training program in an effort to teach business unit managers to write system requirements more effectively. The discovery activity was also often outsourced to consultants or market research organizations (see "Key Themes and Issues" below). As a result, requirements discovery on the part of the design personnel has often become a matter of clarifying and assessing gaps rather than a comprehensive frontal elicitation effort.

**Specification & Modeling.** A central observation with respect to the specification of requirements is that the design professionals did not speak of specific modeling techniques employed. Rather they discussed modeling tools that their design teams use. Requirements management platforms such as IBM's Rational suite and Telelogic DOORS were more salient than the specific types of models developed. Use cases represent one important exception, however. Virtually all interviewees noted that their design teams engage in use case development as a central aspect of their specification activity. For example, one participant observed:

> "So a few of our more senior developers had really gotten on board with UML, use case modeling, and then are becoming fairly good with some of the software tools out there. Some of us use IBM's Rational suite. Some of them are working with a product we're evaluating called Rhapsody from I-Logix. But the intent there is if we can graphically present to the customer and do modeling to decompose a lot of those requirements, that it really helps in that review to catch anything that's missing."

Modeling was often described as "informal," and involved extensive use of natural language narratives, which is consistent with the widespread adoption of use cases. Beyond use cases, several participants reported the use of process or workflow models, as well as data models / E-R diagrams. While UML was implied by the application of Rational software, for example, only a handful of interviewees specifically indicated that some portion of their requirements process is based on UML.

**Verification & Validation.** None of the interviewees noted the adoption of formal approaches to verifications or requirements checking. Specifically, when asked about verifying correctness and assessing the consistency of requirements, the majority noted that this was accomplished through informal review and discussion among the design team. The following quote is characteristic of the responses to this inquiry:

> "Usually I have at least two developers that are familiar with all the systems. If we get new requirements in we'll all do a blind read and then we'll kind of mark it up, say where do you see some holes … and in some cases we'll say, 'Look at page two, item 3.1 and then look at page fifteen item 9.2, it sounds like you're saying A to Z here and you're saying Z to A there.' And sometimes they'll say you're right, maybe it was written by more than one person or they changed their mind and forgot to change in all the places. So we definitely try to go through the entire thing with more

> than one set of eyes on our side looking for inconsistencies or omissions or things that look like they're definitely, at a minimum, confusing."

When moving from verification to validation, a greater degree of formality is observed. In most organizations validation efforts centered on explicit sign-offs by project sponsors and other stakeholders. The stakeholders are expected to review the requirements documentation and acknowledge their acceptance for the design effort to move forward. One challenge noted in this regard is that stakeholders frequently fail to review thoroughly the documentation that is presented to them (due to lack of time or perceived time-to-market demands), and therefore design efforts are allowed to move forward despite the potential presence of significant requirements errors. This phenomenon is exacerbated under conditions of multiple sign-offs because of the diffusion and ambiguity of responsibility.

In addition, the interviews noted frequent use of prototyping, user-interface mockups, and system walkthroughs as validation mechanisms. While none of the organizations represented was extensively involved in agile development, several emphasized iteration and prototype development:

> "To be able to, very rapidly, hear what the requirements are and draft up a prototype, something they can see. Whether it would be, you know, there's varying levels of complexity and money that are associated with something like that right. I could do paper based prototyping or I can do systems based prototyping and having that kind of capability in place to help validate the requirement - is this what you asked for; is this what you would want to see."

Interestingly, a few of the firms indicated novel validation practices, such as validation of use case "personas," validation of time estimates, and stakeholder voting.

### 4.3 Key Themes and Issues

Beyond the state of current practices, we identified a number of recurring themes and issues that were inductively derived from the interview data. These are themes that tap into emerging trends or patterns in the requirements domain. Table 4 summarizes these themes.

It is important to note that these nine identified themes are not mutually exclusive. Indeed, there is significant conceptual affinity among several of the themes. However, the distinctions and level of detail that is presented emerged from the data through multiple rounds of coding among the authors, with a goal of deriving a parsimonious yet thorough representation of distinct themes and constructs in the data. In several cases, study participants proposed causal relationships between themes, but such causal assertions varied significantly and often suggested a recursive pattern (e.g., focus on integration leads to implementation of packaged software and the implementation of packaged software necessitates a focus on integration). Therefore, we will refrain at this stage from making any causal assertions with respect to the themes, but will discuss some of them critically in the discussion section. We will discuss each of these themes in detail after characterizing the general emerging pattern of requirements practices.

**Table 5.** Summary of Key Themes & Issues Associated with Design Requirements

| Requirements Theme | Brief description |
|---|---|
| Business process focus | Requirements process focusing on the business process, and requirements for technological artifact driven by business process. |
| Systems transparency | Requirements driven by demand for a seamless user experience across applications. |
| Integration focus | Requirements efforts focus on integrating existing applications rather than development of new ones |
| Distributed requirements | In addition to diverse stakeholders, requirements process distributed across organizations, geographically, and globally. |
| Layers of requirements | Requirements iteratively developing across multiple levels of abstraction, design focus, or timing. |
| Packaged software | Purchase of commercial off-the-shelf (COTS) software rather then development – trend toward vendor-led requirements. |
| Centrality of architecture | Architectural requirements take a central role, and drive product and application requirements. |
| Interdependent Complexity | While some forms of complexity have been reduced, overall complexity has risen significantly. |
| Fluidity of design | Requirements process accommodates the continued evolution of the artifact after implementation. |

We can describe the emerging practice of requirements as follows: *business process design* take precedence in the design of individual artifacts, and thereby represents a central source of complex socio-technical design requirements. The business process emphasis is driven by an increased demand for *transparency* across distinct systems and the corresponding focus on *integration* over more traditional isolated development efforts. As a result, the requirements process is *distributed* across functional, organizational, and geographic boundaries. The distributed nature of requirements underscores the existence of multiple *layers of requirements*, based on differences in abstraction, user-orientation, and timing. Such layering of requirements is illustrated in the marked emphasis on the use of *commercial-off-the-shelf (COTS)/packaged software* in most of the organizations represented. The heterogeneity of design artifacts within existing business environments in turn necessitates a focus on the adherence to established *information architectures* for all subsequent product and system design efforts. This emphasizes conformance to information *architectures* in guiding requirements, and channeling the implementation of COTS software, rather than developing from scratch. These increase the level of *interdependent complexity*, as well as *layering of requirements* across multiple dimensions. Finally, because of complexity and continuous change, designs are *fluid* as they continue to evolve after implementation, which stresses the importance for requirements to evolve. A more thorough exploration of each of these themes provides an original look into multiple factors that currently drive change in requirements practices.

**Business Process Focus.** One of the distinct insights to emerge from the study is a consistent shift from a focus on a particular application and its associated work practices to a focus on chains of work practices – or business processes – within which a given set of applications is situated. The comprehensive integration of information system components has become more prevalent, driven by the design focus on end-to-end business processes that utilize these technological resources. Accordingly, requirements for specific artifacts increasingly flow from the holistic understanding of the business process itself. This shift was prevalent in many interviews, but it is not as readily apparent in the research literature.[8] One informant describes this shift as follows:

> "The requirements are often based on the business process… Typically what you would do together with the requirements is you would define your business processes. You define the process flow and the main process steps at that point. You make the main activities. And then you can map the requirements to that."

More pointedly, one respondent mapped out the crucial role of business process management to requirements engineering efforts:

> "The rise of Business Process Management may largely affect the RE process in the near future. The [organization] is already running a pilot project which: Generates requirements (AS-IS / TO-BE modeling) through the Business Process Management practice; translates part of these requirements to specifications for the configuration of Workflow Management and Business Rule Management tools; refers to specific services of the SOA [service oriented architecture]. This way of working will not be applicable to all IS projects, but it seems suitable for particular types of applications."

In a similar vein, a trend that a number of informants identified suggested that the boundaries between business processes and IT are becoming increasingly blurred:

> "There's no such thing as an IT project, there's a business project where IT is part of it. And that's been helpful, but at some point, businesses are going to want IT leaders to in effect be innovative in relation to what the next kinds of solutions are. Some people have said that CIOs should become chief process officers."

The logic behind business investments in IT now emphasizes having organizational priorities and processes to drive IT development rather than letting IT capabilities determine the priorities of activities. Since the bursting of the dot com bubble, most organizations have heard this message loud and clear [141, 142].

**Systems Transparency.** The orientation toward the design of business processes implies a movement away from system boundaries based on arbitrary functional or divisional criteria. Business users and consumers alike demand transparency across

[8] An area of notable exception is the study of requirements engineering in workflow automation systems (e.g., [139, 140]).

software applications. Technologies are expected to converge so as to provide a seamless user experience and generate perceptions of a single service platform. As one participant noted, new solutions must be available anywhere, anytime, anyhow and are often expected to be device-independent. The concept of *systems transparency* highlights increased concerns of the users that emphasize the design of a seamless and uniform user experience. While a great deal of traditional requirements literature focuses on the notion of usability associated with a specific application, systems transparency calls attention to unified usability of a portfolio of applications. The requirements process is no longer about a user's interaction with a single application; rather, it is oriented toward the user's overall experience which is situated within an ecology of applications and design artifacts. One informant succinctly captured this idea of systems transparency, by emphasizing a trend toward personalization of applications:

> "The desire from the customer side is much more for applications to cross whatever artificial boundaries exist in terms of data sources and in terms of small systems coming together. I see a big change in what IT does for requirements focusing more on achieving user-centricity and giving people more of a personalized view of how to do their job and get the data going… a user shouldn't have to worry about what device they're using or what system it's in, just getting the information they need when they need that. That's really a major change in how systems are designed … inevitably the end user customers want a seamless integration, they want a common look and feel…"

In order to accomplish such "user-centricity," the requirements process must focus upon work roles and overall activity in order to provide systems that fit within the distinct daily practices of individuals. According to one interview, this focus "really changes IT people from being raw functional application creators, to being more of, you know, performance architects." Naturally, the seamless user experience must be enabled by linking applications which is addressed by the next theme.

**Integration Focus.** *Integration focus* denotes efforts associated with making user experiences possible through integrating applications and system components. While the bulk of the literature on requirements addresses the creation of new applications for specific purposes, many study participants downplayed the importance of *designing* individual artifacts, while emphasizing instead the criticality of integration across applications and capabilities. The focus on integration was one of the most pronounced themes across all interviews. The following statement is emblematic:

> "I'd say that the big difference that we've gone through over the five year period was a transition from one standalone system, which might have lived on individual desktops, or lived on a single network for delivery to departmental users, to now more integrated applications which are tied together via one way or another. Whether it's at a database level or whether it's a web services or whatever, we have applications now that need to share data between systems, between business units, and between our affiliated partners."

While this integration is driven by user considerations, there are host of other organizational drivers that shift requirements practices towards integration:

> "With the tighter integration of supply chains and the customer base, you can't have processes and systems that are not integrated… So that brings together the different applications, the platforms they're built on, the middleware and the data structures that you have to have in place to integrate this stuff. If you don't have it in place you design some islands of functionality that don't work together."

A primary implication of the trend toward integration is that the role of internal IT groups is changing rapidly. Many informants characterized their groups more as integrators than developers: the main proportion of their work is oriented toward assessing and maintaining interdependencies between systems rather than being involved with traditional functional requirements and subsequent design. As one participant put it, "for those of us in IT [the need for integration] changed us from application developers into systems integrators."

**Distributed Requirements.** A pattern that became apparent during the study is the increased distribution of requirements processes across functional, organizational, and geographic boundaries. Frequently, no single organization or functional unit is responsible for the development of the bulk of design requirements. Vendors, consultants, enterprise architects, development teams, business stakeholders, and individual users all play significant roles in articulating and implementing requirements. Furthermore, the widespread adoption of outsourcing has expanded the geographic spread of requirements discovery and development efforts.

Globalization has become a critical force behind much of this distribution, but the implications of globalization go beyond obvious growth in geographical and cultural distance. Distributed requirements bring with them business contexts and features that are tied to distinct locations and business environments. One informant illustrated this vividly:

> "[The organization] represents a large corporation and so you know, we have various initial conditions. So for example, a climate control module might have been supplied by supplier X sitting in Munich and they might have actually inherited along with the hard box, they might have inherited a whole lot of models. Those models would have had certain class definitions. Those class definitions would have been based on somebody else that the supplier was actually supplying some super system to. So we now have to incorporate those classes if we really wanted useful…if it has to be useful to my direct mainstream customer. So we can go create our own universal model here but our customer will have to go through a translation phase."

Not only are requirements distributed geographically, but they are spread across organizations as well. The prevalence of COTS applications (see discussion below) and the growth in industry wide standards results in the significant distribution of requirement's discovery and validation efforts among multiple independent organizations. While this is necessarily the case to an extent for all COTS, it is also amplified

by the complexity of packaged systems and the increasingly limited knowledge many organizations have in formulating solutions:

> "Now we are rolling out my [software system]. That product is way beyond the capabilities of my current team to implement because they haven't spent the months and months of learning how to use this new technology. And we don't have the months and months to wait to train people before we start doing requirements definition and process design."

With this emergence of the distributed nature of design, collaborative tools have become increasingly important for managing requirements that are drawn from increasingly diversified sources. One informant captured the new division of labor that results from distributed requirements as follows:

> "The significant advantage of that is people could be collaborating, not only synchronously but asynchronously … everybody gets to see what the other person is contributing … So for example you might want to have a group of people defining the technical aspect of it. Then you have another group which is defining the business aspect of it. And you have a third group working the external collaborators. And they all have parts and pieces to do it."

Distributed requirements also enhance parallelism in requirements processes. Given the traditional focus on singular individuals or teams managing requirements processes, there is notably little discussion in the literature about the implications of parallel requirements efforts. One area of exception in this regard is a recent focus on the impact of geographically distributed teams engaged in various facets of the requirements process [143-145].

**Layers of Requirements.** Contemporary design efforts generally entail multiple layers of requirements. These layers may be associated with differing levels of abstraction, design focus, user-orientation, or timing. This layering phenomenon includes the traditional transition through business, functional, and technical requirements [10]. It also includes organizing requirements based on the level of analysis. For example, the process for articulating and managing the requirements of enterprise architecture differ from those considered in the development of an individual application:

> "For example, in the situation that I'm currently in with one of my existing clients, we are not only building new applications for them, we're also building application architectures. So there's two sets of requirements; there's business requirements for the different applications that we're building and then in turn what we have to do is for those applications, what are the set of infrastructure requirements that our technology infrastructure team needs to build to be able to provide the framework that these applications will be built on. Whether that be a reporting architecture or a real-time architecture, batch architecture, online architectures, et cetera."

The volatility, or timing of requirements, is another key basis for layering. Requirements that are expected to persist over an extended period of time demand

distinct approaches from requirements that change rapidly, and in less-predictable ways. This phenomenon is relevant in the design of embedded systems and product lines because the requirements volatility (variability) for the embedded artifact in the product differs significantly from that of the underlying system, as the following statement illustrates:

> "In terms of being able to span the feature space if you will, cover it, there's a timeless element to it - people always want some aspect, they want a place to sit down and they want to be able to steer and drive. There's a piece that changes with very slow clock speed, that's the package and physical aspects of it. And then there's the fast cycle, fast clock speed, aspects. So there's really…there's a DC component, there's one that really changes very slowly and there's one that changes very fast."

The emergence of new bases for the layering of requirements has clear affinity with the distribution of requirements. Layers of requirements may be discovered, specified, and managed across distinct stakeholder groups or organizations. Increased challenges are created by shifts to build mechanisms that ensure consistency across different layers.

**Packaged Software Orientation.** For systems design efforts, the interviews depicted a clear preference for using commercial-off-the-shelf (COTS), or packaged, software over the development of separate, new applications. The following quote is a good representative of the sentiments espoused:

> "We made a decision that we were going to pick SAP and purchase it in advance of all these projects going live because, in a fact, we wanted to send a signal to the organization that we've picked SAP and it's the only solution you're going to use going forward."

This represents a major point of departure from much of the requirements research tradition, which, often implicitly, conceptualizes requirements practices as taking place in the context of a greenfield development where the end product is a new software system. Requirements for packaged software implementation projects are significantly different from those of traditional development. For example, software vendors and consultants have a great deal of involvement and often take the lead in requirements processes. The prevalence of packaged software creates a new dynamic for requirements processes. In contrast, to the traditional claim that requirements processes should focus on the "what" of a design effort without respect to "how" it will be achieved [38], the use of COTS implies that much of the "how" is already established at the outset of a requirements effort. In these cases, the requirements process begins more with a "gap" analysis between processes supported by the package and the desired work practices:

> "The easiest one is just to take the software as it comes out of the box, some type of a pre-configured solution that may be industry specific, may not be industry specific. And you run workshops where you sketch out the future processes, walk through the software itself in its current state, and identify any gaps with that process and the software and the requirements you have already defined."

While the prevalence of COTS applications was clear, many of the study participants did reflect on the drawbacks to packaged software – especially in terms of large enterprise vendors, and their firms' dependence on such vendors as a major requirements constraint. However, with the exception of truly unique and critical applications, the benefits of COTS appear to outweigh these drawbacks in the minds of our participants (e.g., lower cost, better predictable quality).

**Centrality of Architecture.** A consistent finding in the study was the growing recognition of the importance of information architectures in establishing the context for requirements processes. In many of the organizations represented, adherence to established information architectures has become a critical concern and constraint for all design efforts. In large part, the specification of formal and encompassing enterprise architectures is driven by the need to address integration complexity and need to maintain consistency in applications and organization wide process designs. Therefore, architectures have become essential for requirements activity and set the baseline constraints for all subsequent designs.

Because the study involved both functional IT units and product development organizations, two types of architectures were salient to design practices: enterprise architecture and product architectures. Many participants indicated that enterprise architectures, especially those associated with an organization's internal IT infrastructure, are becoming critical as organizations look to integrate extensive portfolios of applications that have resulted from previous stove-piped legacy development. Another driver is organizational mergers and acquisitions. To move forward with development projects, an enterprise-level justification and adherence to the established architecture is essential. As a result, architectures precede and drive the requirements of specific artifacts, rather than the requirements driving the development of models:

> "In fact we have a very strict architecture team and an architecture review board all focused in my area on, as projects are begun, to insure that those projects move forward with the long term architecture of [respondent's firm] in mind."
>
> ……
>
> "The architecture determines the scope of application functionality and requirements which you can do. But if you look at the sort of future evolution it may be that you make currently the right architecture choices but maybe two years down the road another requirement emerges and you are stuck."

In some cases, the enterprise architecture represented significant constraints on the requirements processes, while in other cases it just changed the structure of these processes. As a large banking organization indicated:

> "The RE process is being tailored to the special needs of the aforementioned architecture in various ways. For example, business analysts are aware of systemic calls to the core banking system and refer to them in detail when they design business functions. On the other hand, the bank is still on the process of adopting a service-oriented, multi-channel approach. The current, rather immature, situation (as far as multi-channeling is

> concerned) generates a need for separating RE according to the service delivery channel. For example, the collection of requirements for implementing a new module on the core system is differentiated from the collection of requirements for the same module on the e-banking platform."

Similarly, organizations developing products focused increasingly on the development of formal product architectures to support the managed evolution of their offerings. This is particularly important whilst technologies change so rapidly that architects essentially "bet on" standards in order to accommodate unknown future changes in technologies and their demand.

**Fluidity of Designs.** Those interviewed showed an increased appreciation for the fluidity, or continued evolution, of design artifacts. While artifacts have always evolved after use, design teams have traditionally viewed a project as "complete" at some point – normally after software implementation. Informants indicated that this assumption about designs has begun to wane as they recognize that projects often form a single step in an iterative process: "You know as soon as we build a project and deliver it, the day after, we're in the enhancement phase." One strategy for dealing with this evolution was to limit the scope of projects intentionally, with planned and managed releases:

> "You have to set the expectation that what will be delivered will not be exactly what you're looking for. It's not going to be the end; it'll be a step along the way. We are going to build this and then we are going to expand on that capability in subsequent releases. You can't deliver to the end goal right out of the gate…"

Users appropriate artifacts in idiosyncratic ways. Many firms are therefore increasingly cognizant of this evolution and are not attempting to define all requirements ahead of time. Rather they seek to provide additional mechanisms to empower end users to personalize the artifacts. They may build open interfaces to allow evolution in requirements:

> "We're pushing the capability out to the end users and saying don't put us in the middle of it, if you want to figure this out here's the [tool] … the data definition is there, you can select the data that you want to put on the report, how you want it on the report, where you want to gather the data from, what kind of sort sequence, what kind of summary information you want. We're pushing that capability out to the end users so they can self serve just like, you know, companies are pushing capabilities out to their end customers so they can self serve and reduce the cost to serve overall."

In a product development, the challenge is to generate requirements that tolerate "fuzziness," as one product development manager indicated:

> "I don't really understand what the consumers actually prefer and since its change is faster than I can change my [design], how can I design in ways that somebody else can fiddle around with it? … These things are changing so fast it's invention in the hands of the owner, how you design your systems in a way that you make that possible."

Solutions to unknown user-led evolution involved increased reliance on interface standards and standardized "platforms" embedded into products. Rather than specifying specific functional requirements, as these can not be known, standard interfaces that may accommodate multiple add-ons have become the main object of requirements.

**Interdependent Complexity.** The final persistent theme was the perception of increased complexity. This was associated with varying technologies, requirements efforts, and design processes. While it has long been observed that complexity is an *essential* rather than an *accidental* property of systems development [146], most participants felt that the complexity they now encounter has increased significantly:

> "You know, certainly your ability to manage the sheer quantity of requirements and to be able to test those requirements. Usually, on these large complex systems where you're talking about hundreds and hundreds of different applications, your ability to test in an integrated fashion, all of those requirements is very, very hard and very costly."

However, the level at which such complexity emerges has also shifted – from the internal complexity of software development to the integrative complexity of interdependent systems:

> "I have not seen from my area, the complexity be nearly as mammoth as like when we did MRP back in the mid-1980s, where we had hundred and hundreds and hundreds of programs integrated into three hundred batch jobs and all synchronized to run in a 48-hour period for regenerative MRP once a week - not to count the dailys and weeklys. I don't see that the IT projects have that level of complexity in terms of tying things off. What I do see is that the complexity from systems integration and how to secure at the application level the appropriate details, has gotten a lot more complicated."

Despite the deployment of modular designs and architectures, complexity is now substantially greater because of the large number of interdependent systems, increased number of new systems that must be integrated with legacy infrastructure, and the sheer magnitude of integration-oriented requirements given all of the themes. Indeed, complexity in its various manifestations is perhaps the main fundamental motif that cuts across all the issues raised in this study.

## 5 Discussion

The findings from this study pose a series of engaging questions to researchers interested in design requirements phenomena. Introspectively, we must ask ourselves the degree to which the assumptions that underlie current research traditions have inhibited us from understanding and attending to the ways in which requirements work is actually accomplished. Turning to the observed processes and factors emerging in contemporary design practice, we may ask how best to make sense of the diverse forces that affect designers. Finally, we must consider the avenues that are opening up

for productive inquiry around emergent requirements themes and how these are related to existing threads within the research community.

### 5.1 Current Practice and the Research Tradition

Our results point to a great deal of progress associated with requirements practices, but they also signal a changing emphasis towards infrastructural, organizational, and environmental complexity. On the one hand, systems development organizations now employ many of the principles that researchers have been advocating for years, such as formal validation and sign-off, enterprise and functional modeling, user involvement, explicit risk management, and the use of CASE tools. Furthermore, many are looking to increase the degree of structure in their requirements practices. On the other hand, there appear to be a number of inconsistencies between the way practitioners view requirements processes and the way requirements topics are treated in the research community. For example, practitioners do not make many of the distinctions that researchers favor (e.g., phases of the requirements process and requirements types); they often don't refer to formal methodologies and their practices are not consistent with a singular methodology; and, practitioners de-emphasize formal modeling's role in discovery and validation/verification, betraying a continued preference for natural language to capture and communicate requirements.

While our findings reveal that academics may be leading the practitioner community in many respects, they also indicate that some of the assumptions reflected in the literature are not shared by design practitioners. Thus, we must ask ourselves a challenging question – has the practitioner community simply not yet caught up, or are many of our scholarly assumptions not relevant to requirements practice? One of the seemingly more problematic assumptions concerns the distinction between facets in the requirements process. While the research community has long acknowledged the importance of iteration and the interplay of processes in requirements efforts [3], most requirements texts outline a temporal sequence of requirements tasks or phases [e.g., 10, 12, 39]. Furthermore, the structure of the research discourse is clearly bounded by distinct phases, with researchers focusing largely on a single facet of the process (e.g., specification and modeling) in their research efforts. The activities we have labeled as "discovery," "specification," and "validation & verification" [39] have been framed a number of different ways in the literature, yet our interviews indicate that these distinctions are rarely made in practice. Discovery, specification, and validation/verification practices often happen simultaneously, and are largely mediated through natural-language and various artifacts. At the very least, they are so closely related that the practical distinctions between these tasks have become difficult to draw. Iteration continues throughout design, as discovery revolves and verification never really ceases, even after the delivery of a system. Throughout this process, interactions with user communities are conducted through natural language and design artifacts of different maturity.

A second key assumption concerns the estimated value of formal modeling techniques. Formal modeling remains squarely within the development community, and it does not appear to have been appropriated as a communication tool with users.[9]

[9] One exception to this observation is business process flow diagrams that are widely used to mediate developer-user communication (in relevant applications).

Rather, to the extent that formal models are used, they are derived from natural language representations and created after requirements have been generated and approved. The academic literature on modeling seeks to make the natural language communication between users and developers more precise through formal models, but this approach does not appear to be followed by designers themselves. With the significant adoption of use cases, we find that greater precision in designer-user communication is indeed desired, but it is fulfilled through semi-structured natural language exchanges. Thus, we may question the assumption that formal modeling effectively supports interactions between distinct stakeholders or bridges the communicative gaps between the design team and other stakeholders [4, 80, 81, 93, 147]. Perhaps a more appropriate pursuit would be to augment formal models with well organized and structured natural language representations.

In our attempt to re-evaluate assumptions that are challenged by contemporary practice, it is important that we understand the emerging patterns of requirements processes within complex designs. Much of the academic literature is rooted in the paradigm of the 1970s and 1980s, where systems were developed from scratch (in distinct, typically "greenfield" projects) in order to support a specific set of operational activities. Two primary groups were involved: those that would use the artifact and those charged with the development of it. Within this context, it was commonly understood that requirements for the system were in the heads of the users, and it was up to the developers to elicit these requirements to guide further system design. As has been extensively illustrated, this assumption was rife with difficulty, as multiple stakeholder groups were affected by the system, and requirements were rarely easily accessible and had a tendency to change over time while systems evolved (e.g., [146]). In subsequent years, strategies have been put forward to overcome some of these challenges. Techniques such as prototyping [54] and ethnographic methods have been adopted to help designers move beyond the limitations of traditional methods [48, 53]. Yet, these approaches remain well within the traditional paradigm, as they are methods to improve the extraction of requirements from users for increased formalization by developers.

### 5.2 Understanding Emergent Forces in Requirements Practice

Our data suggest a number of different trends that present challenges to the traditional outlook. Systems are no longer created for a specific set of activities within an organization's functional silos. Systems are intended to support cross-organizational and inter-organizational business processes or to offer multiple functions over a product's life-cycle. Multipurpose, expandable devices for a wide array of applications abound. Systems increasingly connect to each other, become integrated, and system boundaries and associated connections are made invisible to users. Greenfield efforts are nearly extinct, and stakeholders are no longer a fixed or easily-identifiable set of individuals. Commercial-off-the-shelf (COTS) applications and modular architectures dominate design landscapes as firms look to buy rather than build due to lower cost and expectation of higher quality. Development never ends. When one level of complexity becomes black-boxed, additional layers of complexity emerge. No longer can we look at the requirements process as a dialogue where designers extract requirements from users. Rather, designers and design teams can be best viewed as reconciling multiple forces and perspectives: negotiating with an ever-expanding set of

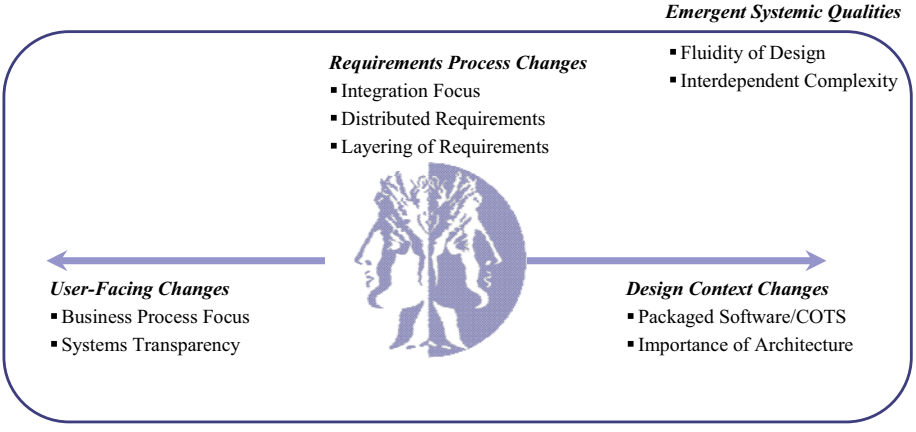


**Fig. 1.** An Emerging Requirements Landscape

stakeholders, merging and connecting evolving architectures, addressing continuing technological changes, and mitigating the complexity associated with marrying these threads. Figure 1 illustrates one attempt to organize and structure the diverse forces that drive this emerging requirements landscape.

This new requirements landscape evokes again the Janus-faced nature of requirements practice. The designer is caught between two fluctuating worlds, where he or she is simultaneously looking backward towards the shifting sands of stakeholder needs, while looking forward to evolving platforms and technological architectures and the concrete design implications that they bear. Within this context, we observe themes that speak to the changing ways in which stakeholders encounter, and interact with, the software-intensive artifacts that populate their work and home environments. These "User-Facing Changes" reflect a shift in expectations on the part of individual users and groups. As they accept novel technologies into more and more facets of their lives, they expect the boundaries between artifacts to fade into the background, minimizing the cognitive effort required to transition between tools within their socio-technical ecology. We assert that the observed themes of *Business Process Focus* and *Systems Transparency* embody these changes in the users' experience.

At the other end of our Janus's line of sight, we observe significant changes in the design contexts within which design teams must maneuver. "Design Context Changes" reflect a fundamental shift in the baseline conditions for all contemporary software-intensive design efforts. The decline of traditional development activities and the critical of contextual constraints have dramatically altered the process of design itself. The rising emphasis on *Packaged Software/COTS* and the centrality of *Information Architectures* are two clear manifestations of this observed shift.

Between the changing expectations of users and the altered constraints on the broader design environment sits the Janus of requirements. In an effort to marry these diverse forces, the process of requirements has itself been transformed. Current requirements practices reflect a more heterogeneous and multi-faceted phenomenon than is often reflected in the treatment by the research community. A significant *Integration Focus*, the management of requirements from a varied set of sources

**Table 6.** Proposed Emergent Avenues for Requirements Research

| Key Themes | Overall | Selected Topics from the Requirements Research Tradition | | |
|---|---|---|---|---|
| | | *Discovery/ Elicitation* | *Specification & Modeling* | *Validation & Verification* |
| ***User-Facing Changes*** | | | | |
| **Business Processes Focus** | Understanding requirements processes when the technology fades into the business context | Assessing the effectiveness of discovery techniques in business process design efforts | Coordinating enterprise, business process, and functional models | Evaluating adherence to strategic business processes |
| **Systems Transparency** | Capturing benefits and challenges posed by the transparency of systems | Determining the sources of user expectations with respect to systems interoperability | Capturing the needs for transparency as a functional requirement in modeling methods | Evaluating new prototyping and simulation capabilities |
| ***Requirements Practice Changes*** | | | | |
| **Integration Focus** | Determining who is responsible for functional requirements when traditional development is less relevant | Determining appropriate stakeholders for articulating integration-oriented requirements | Managing heterogeneous models from a variety of systems | Understanding the processes employed for ensuring satisfactory integration testing |
| **Distributed Requirements** | Effective management of requirements in a distributed cognitive process | Aggregation of requirements identified by multiple parties; Coordinating among different elicitation activities | Managing heterogeneous models from a variety of systems | Understanding validation and verification of aggregated requirements |
| **Layers of Requirements** | Identifying new bases for requirements layering in embedded systems and other contexts | Assessing differences in effectiveness of discovery approaches based on layers observed | Assessing what layers are most amenable to natural language vs. modeling methods | Developing mechanisms for requirements checking across multiple layers |

**Table 6.** (*continued*)

| Key Themes | Overall | Selected Topics from the Requirements Research Tradition | | |
|---|---|---|---|---|
| | | *Discovery/ Elicitation* | *Specification & Modeling* | *Validation & Verification* |
| ***Design Context Changes*** | | | | |
| **Packaged Software Orientation** | Understanding the role of prognostication in requirements - who will be the winner in a given market? | Capturing the potential for knowledge gains through the use of vendors | Marrying current state and future state models to stated vendor preferences | Determining the degree to which COTS address platform-agnostic requirements |
| **Centrality of Architecture** | Identifying the ways in which architecture impacts requirements practice | Observing how architecture sets constraints on discovery processes | Models driving the requirements rather than requirements driving model development | Architecture as the arbiter of appropriateness and quality |
| ***Emergent Systemic Qualities*** | | | | |
| **Fluidity of Design** | Requirements in the context of partial designs<br>Run-time evolution of requirements | Evolution of stakeholder needs based on continued system use | Management of models over generation of design iteration | Managing stakeholder expectations and validation in fluid design contexts |
| **Interdependent Complexity** | Paradoxes of reducing and increasing complexity at different levels of analysis | Determining appropriate levels of stakeholder involvement in complex design efforts | Managing heterogeneous models from a variety of systems | Requirements testing methods for application in highly-interdependent environments |

(i.e., *Distributed Requirements*), and the emergence of novel bases for the *Layering of Requirements* all embody the changes to be observed in modern requirements processes.

In addition to the changes observed in user experiences, design contexts, and requirements processes, there are broader contextual characteristics that have emerged from, and in turn engendered, the other themes we have highlighted here. These "Emergent Systemic Qualities" call our attention to important new areas for exploration and rigorous inquiry. The rise of *Interdependent Complexity* is a force that can be seen at all levels of the design process – from the expectations of users to the practical, technical demands reflected in novel artifacts. As designers have struggled to control complexity at one level of work, it has re-emerged at another level. Complexity has become an intrinsic part of design and affects both stakeholder behaviors and other extrinsic forces. Similarly, the recognition of the *Fluidity of Design* in the organizations that participated in this study suggests a new maturity of understanding with respect to the impermanence and evolutionary potential that is central to modern software-intensive systems design.

A shifting focus toward integration and evolution rather than elicitation and documentation highlights the increasingly creative role that designers must play in actively co-producing requirements and artifacts, rather than simply charting out needs that are "out there" *a priori*. This observation has multiple implications for design research and calls for an expansion of the realm of requirements research to address broader organizational aspects of design and the requirements processes.

### 5.3 New Avenues for Requirements Research

Perhaps most importantly for the present discussion, the observations and phenomena presented in this study call our attention to a wide array of new avenues for requirements research. Each of the key findings reflects an issue that warrants additional exploration. To close the research-practice gap within the requirements arena, some of the more counter-intuitive (and contra-prescriptive) findings from the assessment of current practice should be investigated. Similarly, each of the key themes that emerged from the analysis should be thoroughly examined to improve our understanding of how these forces are shaping today's design environments. The current volume is intended to initiate just such an agenda-setting perspective. Several of the key themes noted in this study are explored in greater depth and nuance in the chapters of this book. For example, a focus on business processes, the fluidity of contemporary design, and the challenges of interdependent complexity are considered repeatedly throughout the volume. Other facets of the emergent requirements landscape have yet to be approached. For example, the distributed nature of requirements processes and sources, the role of centralized architectures in both addressing and driving requirements efforts, and the demands for greater systems transparency promise fertile ground for research in the coming years.

The challenge, of course, is determining how we can draw upon the research tradition while remaining open to the new phenomena at hand. We have suggested that some of the fundamental assumptions that undergird the requirements literature may need to be reconsidered as we look to the future, but how can we effectively leverage the work that has come before this point?

In the opinion of the research team, the framework provided by the extant research may still provide a useful lens for directing the attention of researchers. While distinctions between facets of the requirements process have blurred, the fundamental concerns upon which they are built remain: What does the design need to have (Discovery)? How can we render an unspoken vision in an explicit form around which multiple individuals and groups can gravitate (Specification)? How can we as designers know when we are on the right track and persuade others of the same (Validation & Verification)? Each of these high-level conceptual challenges offers a perspective that can be fruitfully applied to the emergent phenomena observed among practicing design teams. In Table 5 (provided in the Appendix), we illustrate how traditional requirements research foci and the key themes outlined in this study can be combined to open up a wide range of prospective channels for requirements research in the coming years. Clearly, the issues presented in the table are far from exhaustive, as they are intended merely to illustrate the types of inquiries that are suggested by the framework.

## 6 Conclusion

In this study, we have reflected upon the degree to which the literature on requirements appropriately reflects current design practices across a variety of organizations. We find that recent decades have seen a significant amount of progress in orienting designers to many critical requirements-based considerations, but we also observe a number of issues where current practices are less than consistent with the assumptions of the academic literature. Moreover, we identify a number of macro-level emerging trends across a variety of modern requirements practices. We conclude with a characterization of complex large-scale requirements efforts as an exercise in balancing constraints and opportunities in multiple directions at the same time. Designers, like the Roman god Janus, must simultaneously look to the often-ambiguous needs of stakeholders and attend to the practical demands of the design environment. In so doing, they have changed the face of requirements practice, and ushered in a period of expanding complexity and evolutionary dynamics in design. Contemporary designers construct requirements in relation to existing systems and practices, rather than simply eliciting them as much of the literature implies.

Using this empirical research as a backdrop, we now embark on an effort to make sense of these issues. In a series of two workshops over two years, some of the world's top thinkers on requirements issues will be assembled to discuss the implications of our findings, to address issues we have not yet considered, and to broadly assess the direction of requirements research going forward. During this time we plan to assess the current practices of both the research and the practitioner communities in light of emerging trends in requirements activity, technologies, and organizational environments, with an eye to the following questions: Is the way researchers think about requirements adequate going forward? Do our assumptions about requirements practices need to change? Where should research into requirements focus or expand to capture emerging trends in system design? Thus, the current study is intended as food for thought. We are confident that tremendous insights lie ahead.

**Acknowledgements.** We are grateful for the interviewees for their time, insight, and enthusiasm around the research topic. We are also grateful for constructive feedback and criticism from John Mylopoulos, Peri Loucopoulos, and Bill Robinson. Normal caveats apply.

## References

1. Simon, H.: The Sciences of the Artificial. The MIT Press, Cambridge (1996)
2. Norman, D.A.: The Design of Everyday Things. Basic Books (2002)
3. Nuseibeh, B., Easterbrook, S.: Requirements Engineering: A Roadmap. In: Proceedings of the conference on The future of Software engineering, pp. 35–46. ACM Press, New York (2000)
4. Ross, D.T., Schoman Jr., K.E.: Structured Analysis for Requirements Definition. IEEE Transactions on Software Engineering 3, 6–15 (1977)
5. Guinan, P.J., Cooprider, J.G., Faraj, S.: Enabling Software Development Team Performance During Requirements Definition: A Behavioral Versus Technical Approach. Information Systems Research 9, 101–125 (1998)
6. Mumford, E.: Effective Systems Design and Requirements Analysis: The ETHICS Approach. Macmillan, Basingstoke (1995)
7. Montazemi, A., Conrath, D.: The Use of Cognitive Mapping for Information Requirements Analysis. MIS Quarterly 10, 45–56 (1986)
8. Davis, G.: Strategies for Information Requirements Determination. IBM Systems Journal 21, 4–30 (1982)
9. Yadav, S., Ralph, R., Akemi, T., Rajkumar, T.: Comparison of analysis techniques for information requirement determination. Communications of the ACM 31, 1090–1097 (1988)
10. Wiegers, K.: Software Requirements. Microsoft Press, Redmond (1999)
11. Crowston, K., Kammerer, E.: Coordination and Collective Mind in Software Requirements Development. IBM Systems Journal 37, 227–246 (1998)
12. Sommerville, I., Sawyer, P.: Requirements Engineering: A Good Practice Guide. John Wiley & Sons, Inc., New York (1997)
13. Kotonya, G., Sommerville, I.: Requirements Engineering: Processes and Techniques. John Wiley & Sons, New York (1998)
14. Hoffer, J., George, J., Valacich, J.: Modern Systems Analysis and Design. Prentice Hall, Upper Saddle River (2001)
15. Whitten, J.L., Bentley, L.D.: Systems Analysis and Design Methods. McGraw-Hill/Irwin, New York (2007)
16. Ulrich, K., Eppinger, S.: Product design and development. McGraw-Hill, New York (1995)
17. van Lamsweerde, A.: Requirements Engineering in the Year 00: A Research Perspective. In: Proceedings of the 22nd international conference on Software engineering, pp. 5–19 (2000)
18. Zave, P.: Classification of research efforts in requirements engineering. ACM Computing Surveys 29, 315–321 (1997)
19. Bell, T., Thayer, T.: Software requirements: Are they really a problem? In: Proceedings of the 2nd international conference on Software engineering, pp. 61–68 (1976)
20. Boehm, B.: Software Engineering Economics. Prentice Hall PTR, Upper Saddle River (1981)

21. Boehm, B., Papaccio, P.: Understanding and controlling software costs. IEEE Transactions on Software Engineering 14, 1462–1477 (1988)
22. Scharer, L.: Pinpointing Requirements. Datamation 27, 139–151 (1981)
23. Orr, K.: Agile requirements: opportunity or oxymoron? Software, IEEE 21, 71–73 (2004)
24. Bergman, M., King, J., Lyytinen, K.: Large-Scale Requirements Analysis Revisited: The need for Understanding the Political Ecology of Requirements Engineering. Requirements Engineering 7, 152–171 (2002)
25. Lindquist, C.: Fixing the Requirements Mess. CIO Magazine, 52–60 (2005)
26. Group, T.S.: The CHAOS Report. West Yarmouth, MA (1995)
27. Aurum, A., Wohlin, C.: Requirements Engineering: Setting the Context. In: Aurum, A., Wohlin, C. (eds.) Engineering and Managing Software Requirements, pp. 1–15. Springer, Berlin (2005)
28. Leffingwell, D., Widrig, D.: Managing Software Requirements: A Unified Approach. Addison-Wesley Professional, Reading (1999)
29. Hickey, A., Davis, A.: Elicitation technique selection: how do experts do it? In: Proceedings of 11th IEEE International on Requirements Engineering Conference, pp. 169–178 (2003)
30. Keil, M., Cule, P.E., Lyytinen, K., Schmidt, R.C.: A framework for identifying software project risks. Communications of the ACM 41, 76–83 (1998)
31. Kaindl, H., Brinkkemper, S., Bubenko Jr., J.A., Farbey, B., Greenspan, S.J., Heitmeyer, C.L., Leite, J.C.S.P., Mead, N.R., Mylopoulos, J., Siddiqi, J.: Requirements Engineering and Technology Transfer: Obstacles, Incentives and Improvement Agenda. Requirements Engineering 7, 113–123 (2002)
32. Zowghi, D., Coulin, C.: Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. Engineering and Managing Software Requirements (2005)
33. Siddiqi, J., Shekaran, M.C.: Requirements Engineering: The Emerging Wisdom. IEEE Software 13, 15–19 (1996)
34. Berry, D.M., Lawrence, B.: Requirements Engineering. IEEE Software 15, 26–29 (1998)
35. Zave, P., Jackson, M.: Four Dark Corners of Requirements Engineering. ACM Transactions on Software Engineering and Methodology 6, 1–30 (1997)
36. Wiegers, K.E.: Read My Lips: No New Models! IEEE Software 15, 10–13 (1998)
37. Davis, A.M., Hickey, A.M.: Requirements Researchers: Do We Practice What We Preach? Requirements Engineering 7, 107–111 (2002)
38. Davis, A.M.: Software Requirements: Objects, Functions, and States. Prentice-Hall, Upper Saddle River (1993)
39. Loucopoulos, P., Karakostas, V.: System Requirements Engineering. McGraw-Hill, Inc., New York (1995)
40. Hull, E., Jackson, K., Dick, J.: Requirements Engineering. Springer, London (2005)
41. Jackson, M.: Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices. ACM Press/Addison-Wesley Publishing Co., New York (1995)
42. Windle, D.R., Abreo, L.R.: Software Requirements Using the Unified Process: A Practical Approach. Prentice Hall PTR, Upper Saddle River (2002)
43. Wieringa, R.J.: Requirements Engineering: Frameworks for Understanding. John Wiley & Sons, Inc., New York (1996)
44. Vessey, I., Conger, S.A.: Requirements Specification: Learning Object, Process, and Data Methodologies. Communications of the ACM 37, 102–113 (1994)
45. Dorfman, M.: Software Requirements Engineering. In: Thayer, R.H., Dorfman, M. (eds.) Software Requirements Engineering, pp. 7–22. IEEE Computer Society Press, Los Alamitos (1997)

46. Ghezzi, C., Jazayeri, M., Mandrioli, D.: Fundamentals of Software Engineering. Prentice Hall, Englewood Cliffs (1991)
47. Boehm, B.W.: Verifying and validating software requirements and design specifications. IEEE Software 1, 75–88 (1984)
48. Goguen, J., Linde, C.: Techniques for Requirements Elicitation. Requirements Engineering 93, 152–164 (1993)
49. Agarwal, R., Tanniru, M.T.: Knowledge Acquisition Using Structured Interviewing: An Empirical Investigation. Journal of Management Information Systems 7, 123–140 (1990)
50. Wright, G., Ayton, P.: Eliciting and modelling expert knowledge. Decision Support Systems 3, 13–26 (1987)
51. Byrd, T., Cossick, K., Zmud, R.: A Synthesis of Research on Requirements Analysis and Knowledge Acquisition Techniques. MIS Quarterly 16, 117–138 (1992)
52. Beyer, H., Holtzblatt, K.: Apprenticing with the customer. Communications of the ACM 38, 45–52 (1995)
53. Viller, S., Sommerville, I.: Social Analysis in the Requirements Engineering Process: From Ethnography to Method. In: International Symposium on Requirements Engineering. IEEE, Limerick, Ireland (1999)
54. Alavi, M.: An Assessment of the Prototyping Approach to Information Systems Development. Communications of the ACM 27, 556–563 (1984)
55. Glass, R.L.: Searching for the Holy Grail of Software Engineering. Communications of the ACM 45, 15–16 (2002)
56. Hickey, A.M., Davis, A.M.: A Unified Model of Requirements Elicitation. Journal of Management Information Systems 20, 65–84 (2004)
57. Boland, R.J.: The process and product of system design. In: Management Science, INFORMS, Institute for Operations Research, vol. 24, p. 887 (1978)
58. Ropponen, J., Lyytinen, K.: Components of software development risk: how to address them? A project manager survey. IEEE Transactions on Software Engineering 26, 98–112 (2000)
59. Jørgensen, M., Sjøberg, D.I.K.: The Impact of Customer Expectation on Software Development Effort Estimates. International Journal of Project Management 22, 317–325 (2004)
60. Cook, S., Brown, J.: Bridging Epistemologies: The Generative Dance between Organizational Knowledge and Organizational Knowing. Organization Science 10, 381–400 (1999)
61. Stewart, D., Shamdasani, P.: Focus Groups: Theory and Practice. Sage Publications, Newbury Park (1990)
62. Boehm, B.: A spiral model of software development and enhancement. ACM SIGSOFT Software Engineering Notes 11, 22–42 (1986)
63. Boehm, B.: Verifying and validating software requirements and design specifications. Software risk management table of contents, 205–218 (1989)
64. Garfinkel, H.: The origins of the term 'ethnomethodology'. Ethnomethodology, 15–18 (1974)
65. Lynch, M.: Scientific practice and ordinary action: ethnomethodology and social studies of science. Cambridge University Press, Cambridge (1993)
66. Siddiqi, J.: Challenging universal truths of requirements engineering. IEEE Software 11, 18–19 (1994)

67. Shekaran, C., Garlan, D., Jackson, M., Mead, N.R., Potts, C., Reubenstein, H.B.: The Role of Software Architecture in Requirements Engineering. In: First International Conference on Requirements Engineering, pp. 239–245. IEEE Computer Society Press, San Diego (1994)
68. Baldwin, C.Y., Clark, K.B.: Design Rules, The Power of Modularity, vol. 1. MIT Press, Cambridge (2000)
69. Palmer, J.D.: Traceability. In: Thayer, R.H., Dorfman, M. (eds.) Software Requirements Engineering. IEEE Computer Society Press, Los Alamitos (1997)
70. Ramesh, B.: Factors influencing requirements traceability practice. Communications of the ACM 41, 37–44 (1998)
71. Wieringa, R.J.: An Introduction to Requirements Traceability. Faculty of Mathematics and Computer Science, Vrije Universiteit 1995
72. Hsia, P., Davis, A.M., Kung, D.C.: Status Report: Requirements Engineering. IEEE Software 10, 75–79 (1993)
73. van Lamsweerde, A.: Formal Specification: A Roadmap. In: Conference on the Future of Software Engineering, pp. 147–159. ACM Press, Limerick (2000)
74. Balzer, R., Goldman, N., Wile, D.: Informality in Program Specifications. IEEE Transactions on Software Engineering 4, 94–103 (1978)
75. Ackerman, M., Hadverson, C.: Reexamining organizational memory. Communications of the ACM 43, 58–64 (2000)
76. Reubenstein, H.B., Waters, R.C.: The Requirements Apprentice: Automated Assistance for Requirements Acquisition. IEEE Transactions on Software Engineering 17, 226–240 (1991)
77. Gervasi, V., Nuseibeh, B.: Lightweight validation of natural language requirements. Software Practice and Experience 32, 113–133 (2002)
78. Goldin, L., Berry, D.M.: AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation. Automated Software Engineering 4, 375–412 (1997)
79. Ryan, K.: The role of natural language in requirements engineering. In: IEEE International Symposium on Requirements Engineering, San Diego, CA, pp. 240–242 (1993)
80. Borgida, A., Greenspan, S., Mylopoulos, J.: Knowledge Representation as the Basis for Requirements Specifications. IEEE Computer 18, 82–91 (1985)
81. Mylopoulos, J.: Information Modeling in the Time of the Revolution. Information Systems 23, 127–155 (1998)
82. Checkland, P.: Systems Thinking, Systems Practice. John Wiley & Sons, Inc., New York (1981)
83. Checkland, P., Scholes, J.: Soft Systems Methodology in Action. John Wiley & Sons, Inc., New York (1990)
84. Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language Reference Manual. Addison-Wesley Longman, Essex (1998)
85. Scholz-Reiter, B., Stickel, E.: Business Process Modelling. Springer, Secaucus (1996)
86. Scheer, A.-W.: ARIS–Architecture of Integrated Information Systems. Springer, Heidelberg (1992)
87. Leppänen, M.: An Ontological Framework and a Methodical Skeleton for Method Engineering: A Contextual Approach. University of Jyväskylä (2005)
88. Machado, R.J., Ramos, I., Fernandes, J.M.: Specification of Requirements Models. In: Aurum, A., Wohlin, C. (eds.) Engineering and Managing Software Requirements, pp. 47–68. Springer, Berlin (2005)

89. Mylopoulos, J., Chung, L., Yu, E.: From Object-Oriented to Goal-Oriented Requirements Analysis. Communications of the ACM 42, 31–37 (1999)
90. Brand, D., Zafiropulo, P.: On Communicating Finite-State Machines. Journal of the ACM 30, 323–342 (1983)
91. Peterson, J.L.: Petri Nets. ACM Computing Surveys 9, 223–252 (1977)
92. Harel, D.: Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming 8, 231–274 (1987)
93. Ross, D.T.: Structured Analysis (SA): A Language for Communicating Ideas. Transactions on Software Engineering 3, 16–34 (1977)
94. DeMarco, T.: Structured Analysis and System Specification. Prentice-Hall, Englewood Cliffs (1979)
95. Yourdon, E., Constantine, L.L.: Structured Design. Prentice-Hall, Englewood Cliffs (1979)
96. Sutcliffe, A.G.: Object-oriented systems development: survey of structured methods. Information and Software Technology 33, 433–442 (1991)
97. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: Object-Oriented Modeling and Design. Prentice-Hall, Inc., Englewood Cliffs (1991)
98. Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G.: Object-Oriented Software Engineering: A Use Case Driven Approach, vol. 524. Addison-Wesley, Reading (1992)
99. Bonabeau, E.: Agent-Based Modeling: Methods and Techniques for Simulating Human Systems. Proceedings of the National Academy of Sciences 99, 7280–7287 (2002)
100. Axelrod, R.M.: The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration. Princeton University Press, Princeton (1997)
101. Carley, K., Krackhardt, D.: Cognitive inconsistencies and non-symmetric friendship. Social Networks 18, 1–27 (1996)
102. Carley, K.M.: Computational and mathematical organization theory: Perspective and directions. Computational & Mathematical Organization Theory 1, 39–56 (1995)
103. Epstein, J.M., Axtell, R.: Growing Artificial Societies: Social Science from the Bottom Up. Brookings Institution Press, Washington (1996)
104. van Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Guided Tour. In: Proceedings of the 5th IEEE International Symposium on Requirements Engineering, pp. 249–263. IEEE, Toronto (2001)
105. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-Directed Requirements Acquisition. Science of Computer Programming 20, 3–50 (1993)
106. van Lamsweerde, A., Darimont, R., Letier, E.: Managing conflicts in goal-driven requirements engineering. IEEE Transactions on Software Engineering 24, 908–926 (1998)
107. Mylopoulos, J., Chung, L., Nixon, B.: Representing and using nonfunctional requirements: a process-oriented approach. IEEE Transactions on Software Engineering 18, 483–497 (1992)
108. Miller, J.A., Sheth, A.P., Kochut, K.J.: Perspectives in Modeling: Simulation, Database and Workflow. In: Chen, P.P., Akoka, J., Kangassalu, H., Thalheim, B. (eds.) Conceptual Modeling. LNCS, vol. 1565, pp. 154–167. Springer, Heidelberg (1999)
109. Menzel, C., Mayer, R.J.: The IDEF Family of Languages. Handbook on Architectures of Information Systems 1, 209–241 (1998)
110. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison Wesley Longman, Redwood City (1999)
111. Kobryn, C.: UML 2001: a standardization odyssey. Communications of the ACM 42, 29–37 (2001)

112. Welke, R.J., Kumar, K.: Method engineering: a proposal for situation-specific methodology construction. In: Cotterman, Senn (eds.) Systems Analysis and Design: A Research Agenda, pp. 257–268. Wiley, Chichester (1992)
113. Harmsen, F., Brinkkemper, S., Oei, H.: Situational Method Engineering for Information System Project Approaches. IFIP Transactions A: Computer Science & Technology, 169–194 (1994)
114. Gray, J., Bapty, T., Neema, S., Tuck, J.: Handling crosscutting constraints in domain-specific modeling. Communications of the ACM 44, 87–93 (2001)
115. Ledeczi, A., Bakay, A., Maroti, M., Volgyesi, P., Nordstrom, G., Sprinkle, J., Karsai, G.: Composing domain-specific design environments. Computer 34, 44–51 (2001)
116. Rossi, M., Ramesh, B., Lyytinen, K., Tolvanen, J.-P.: Managing Evolutionary Method Engineering by Method Rationale. Journal of the AIS 5 (2004)
117. Pohl, K.: Requirement Engineering: An overview. Encyclopedia of Computer Science and Technology. Marcel Dekker, New York (1996)
118. Easterbrook, S.: Handling Conflict between Domain Descriptions with Computer-Supported Negotiation. Knowledge Acquisition 3, 255–289 (1991)
119. Grünbacher, P., Seyff, N.: Requirements Negotiation. In: Aurum, A., Wohlin, C. (eds.) Engineering and Managing Software Requirements, pp. 143–162. Springer, Berlin (2005)
120. Berander, P., Andrews, A.: Requirements Prioritization. In: Aurum, A., Wohlin, C. (eds.) Engineering and Managing Software Requirements, pp. 69–94. Springer, Berlin (2005)
121. Regnell, B., Höst, M., Dag, J.N.: An Industrial Case Study on Distributed Prioritisation in Market-Driven Requirements Engineering for Packaged Software. Requirements Engineering 6, 51–62 (2001)
122. Feather, M.S., Fickas, S., Finkelstein, A., van Lamsweerde, A.: Requirements and Specification Exemplars. Automated Software Engineering 4, 419–438 (1997)
123. Robinson, W., Volkov, V.: Supporting the Negotiation Life Cycle. Communications of the ACM 41, 95–102 (1998)
124. Robinson, W.N.: Negotiation behavior during requirement specification. In: Proceedingsof the 12th International Conference on Software Engineering, pp. 268–276. IEEE Computer Society Press, Nice (1990)
125. Fisher, R., Ury, W.: Getting to Yes: Negotiating without Giving In, Boston, MA (1991)
126. Boehm, B.W., Egyed, A.: WinWin Requirements Negotiation Processes: A Multi-Project Analysis. In: 5th International Conference on Software Processes (1998)
127. Grünbacher, P., Briggs, R.O.: Surfacing Tacit Knowledge in Requirements Negotiation: Experiences using EasyWinWin. In: 34th Hawaii International Conference on System Sciences. IEEE, Los Alamitos (2001)
128. Robinson, W.N., Fickas, S.: Automated Support for Requirements Negotiation. In: AAAI 1994 Workshop on Models of Conflict Management in Cooperative Problem Solving, pp. 90–96 (1994)
129. Kotonya, G., Sommerville, I.: Requirements Engineering with Viewpoints. Software Engineering Journal 11, 5–18 (1996)
130. Leite, J.C.S.P., Freeman, P.A.: Requirements Validation through Viewpoint Resolution. IEEE Transactions on Software Engineering 17, 1253–1269 (1991)
131. Knight, J.C., Myers, E.A.: An Improved Inspection Technique. Communications of the ACM 36, 51–61 (1993)
132. Yourdon, E.: Structured Walkthroughs. Yourdon Press, Upper Saddle River (1989)
133. Soni, D., Nord, R., Hofmeister, C.: Software Architecture in Industrial Applications. In: 17th International Conference on Software Engineering, pp. 196–207. ACM Press, New York (1995)

134. Glaser, B.G., Strauss, A.L.: Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine Publishing Company, Chicago (1967)
135. Eisenhardt, K.: Building Theories from Case Study Research. Acad. Manage. Rev. 14, 532–550 (1989)
136. Lyytinen, K., Mathiassen, L., Ropponen, J.: Attention Shaping and Software Risk - A Categorical Analysis of Four Classical Risk Management Approaches. Information Systems Research 9, 233–255 (1998)
137. Kruchten, P.: The Rational Unified Process: An Introduction. Pearson Education, Boston (2003)
138. Vessey, I., Sravanapudi, A.P.: CASE tools as collaborative support technologies. Communications of the ACM 38, 83–95 (1995)
139. Stohr, E., Zhao, J.: Workflow Automation: Overview and Research Issues. Information Systems Frontiers 3, 281–296 (2001)
140. Becker, J., Rosemann, M., von Uthmann, C.: Guidelines of Business Process Modeling. In: van der Aalst, W., Desel, J., Oberweis, A. (eds.) Business Process Management - Models, Techniques, and Empirical Studies, pp. 30–49. Springer, Heidelberg (2000)
141. King, W.R.: IT Capabilities, Business Processes, and Impact on the Bottom Line. In: Brown, C.V., Topi, H. (eds.) IS Management Handbook. Auerbach Publications, Boca Raton (2003)
142. Brynjolfsson, E., Hitt, L.M.: Beyond Computation: Information Technology, Organizational Transformation and Business Performance. In: Malone, T.W., Laubacher, R., Scott Morton, M.S. (eds.) Inventing the organizations of the 21st century, pp. 71–99. MIT Press, Cambridge (2003)
143. Jarke, M., Pohl, K.: Requirements Engineering in 2001 (Virtually) Managing a Changing Reality. Software Engineering Journal 9, 257–266 (1994)
144. Damian, D., Eberlein, A., Shaw, M., Gaines, B.: An exploratory study of facilitation in distributed requirements engineering. Requirements Engineering 8, 23–41 (2003)
145. Grünbacher, P., Braunsberger, P.: Tool Support for Distributed Requirements Negotiation. Cooperative methods and tools for distributed software processes, 56–66 (2003)
146. Brooks, F.P.: No Silver Bullet: Essence and Accidents in Software Engineering. IEEE Computer 20, 10–19 (1987)
147. Greenspan, S., Mylopoulos, J., Borgida, A.: On Formal Requirements Modeling Languages: RML Revisited. In: Proceedings of the 16th International Conference on Software Engineering (ICSE-16), Sorrento, Italy, pp. 135–147 (1994)