Computing Requirements: Cognitive Approaches to Distributed Requirements Engineering

Sean W. Hansen
Rochester Institute of Technology
shansen@saunders.rit.edu

William N. Robinson
Georgia State University
wrobinson@cis.gsu.edu

Kalle J. Lyytinen
Case Western Reserve University
kalle@case.edu

Abstract

We present a study of on the goal-oriented modeling of RE processes executed by a practicing systems development team. The research combines an empirical case study of RE practices with the evaluation and simulation capability of i modeling. Our analysis focuses on a system implementation project at a mid-size U.S. university and applies the theory of distributed cognition to generate a range of design insights for goal identification and process enhancement.*

1. Introduction

Since its inception in the 1970s, the bulk of requirements engineering (RE) research has focused on mechanisms for requirements specification – documenting and formally encoding software requirements. This orientation emphasizes the generation of models that are complete, consistent, and which support downstream design tasks. Despite this persistent focus, RE has remained challenged facet of software development, engendering significant impediments to project success and failing to live up to its original expectations [3, 20, 53].

Importantly, the strong orientation towards specification correctness and completeness is closely tied to traditional sequential development approaches. The recent success of less conventional methodologies, most notably in open source software development [OSSD; 48] and agile development [21, 39] highlight the potential of approaches that rely on natural language, lightweight documentation, constant intertwining of requirements and design, and ongoing discussions of requirements for bringing about better outcomes.

In this article, we suggest that one of the pitfalls of the traditional approaches to RE is that scholars have rarely analyzed how requirements actually get computed as an ongoing cognitive activity within a complex and distributed socio-technical system. Furthermore, researchers have given meager attention to how properties of such a cognitive system either impede or enable 'effective' computation of requirements for the artifact being designed. In this paper we seek to address this oversight by proposing a socio-technical model to analyze and account for RE as a distributed cognitive process. We base our treatment primarily on Hutchin's [24] theory of distributed cognition. We contend that in any RE undertaking a socio-technical system consisting of people and artifacts engages in an ongoing computation of a varying requirement set through tasks of requirements identification, specification, negotiation, and validation.

In developing a distributed process model we outline how the set of requirements is determined using a set of requirements-oriented meta-goals (i.e., organizational goals, general RE goals, and the goals of effective distributed cognitive work environments), the system properties that are represented in a requirement sets, and the goals for a system design process. We outline how the satisfaction of goals at different levels can be analyzed using goal-oriented i* models [60, 61]. These models enable us to determine how specific meta-requirements for RE systems can be derived for an effective requirements computation process based on the theory of distributed cognition. We use scenarios from a relatively complex RE case study involving the adoption, modification, and implementation of an ERP system within a university context. Our analysis demonstrates how the use of socio-technical computational model can help identify pitfalls in the ongoing process and reveal reasons for observed positive and less positive outcomes. Based on this analysis we discuss reasons why open source or agile development processes may experience higher success rates in light of the developed model of requirements computation. Several recommendations for future research on requirements engineering are outlined.

978-0-7695-4525-7/12 $26.00 © 2012 IEEE
DOI 10.1109/HICSS.2012.172

2. Requirements Engineering

For the present study, two facets of RE research warrant a brief discussion: 1) a general overview of approaches to RE, and 2) goal-oriented RE. We offer therefore background of RE research for engaging in the present discussion on RE tasks and goal oriented modeling. We introduce next theoretical foundations related to distributed cognition used in this study.

2.1. Traditional Approaches to RE

A wide array of textbooks and reviews are available, advising practitioners on the effective approaches to RE [e.g., 11, 23, 29, 50]. By comparison, a relatively small percentage of the literature has focused on advancing a theoretical or empirical understanding of how requirements are discovered, defined, negotiated, and managed, and why these processes are so difficult [30]. Moreover, the prescriptive modeling and process methodologies have seldom been subjected to empirical scrutiny [55].

As Hansen et al. [19] note, most research approaches to RE betray a number of assumptions about requirements and the role stakeholders. These assumptions include beliefs that 1) RE facets (e.g., elicitation, specification) can be distinguished in an unproblematic manner, 2) distinct information system components and functionality can be readily delineated, 3) ISD projects are time-bounded efforts for the creation of an artifact that is "complete" at some point in time, and 4) designers can be regarded as an outside party in the application domain [19].

To illustrate the tenacity of these assumptions underlying established approaches to RE, we next briefly review the ways in which the field has been segmented. Just as RE represents one facet of ISD, so too have RE processes been divided into a number of distinct discourses. While researchers have posited anywhere from two to seven primary requirements tasks [12], a widely-employed categorization of the RE process suggests three core facets: 1) elicitation, 2) specification, and 3) validation & verification [38]. *Requirements elicitation* is generally framed as the first component of a design effort – the process by which a designer determines what organizational or customer needs must be addressed by the proposed artifact [16, 33, 38]. While a variety of terms (e.g., discovery, determination, identification) has been used to indicate this facet of the design, the label of *elicitation* is most commonly employed, reflecting the assumption that knowledge about requirements fundamentally rests with users and must be teased out by the designer.

Requirements specification is the process by which the design team acquires, abstracts, and represents the requirements for a design effort [38, 55]. In this context, *modeling* refers to the creation of abstracted representations of the real world through the use of established and formalized symbol systems [41]. The resulting specification represents a transition point where the needs of stakeholders are extended with functional and technical implications that flow from them. A specification must support ease of interpretation and understanding by all stakeholders, while presenting a sufficient foundation for the subsequent technical development. No subject has received more attention within the RE literature than requirements modeling and formulation of formal notation techniques for the specification [53]. In fact, it has been argued that modeling lies at the heart of the RE undertaking [5].

Requirements validation and verification ensure that the requirements are 1) of high quality, 2) address the users' needs, 3) are appropriate for the design effort, and 4) have no inconsistencies or errors [4]. Validation and verification address questions of whether or not the designers have conducted the RE processes effectively and the degree to which the specifications will support a productive design effort.

Through the strict segmentation of requirements processes and a focus on related solutions RE researchers have tended to promote a techno-centric approach to RE tasks [37, 49]. For example, during specification the primacy of the designer's perspective on the development process has been reinforced [19]. This has led to a heavy emphasis on formal notation systems and modeling approaches within the RE research [53].

2.2. Goal-oriented RE

A recent extension to the development of modeling frameworks has been to relate approaches or to formulate modes that describe, guide, or organize RE tasks using system ontologies, notations, processes, or goals [31]. An important element in this trend is the emergence of goal-oriented requirements engineering (GORE) [46, 43, 62, 10, 52]. The GORE research focuses on modeling the objectives, or *goals* under which a system development effort is undertaken [52]. Consequently, several GORE languages have been developed such as *i** [59, 61], GRL [60, 1], KAOS [54, 10], and Tropos [7, 42]. A GORE approach allows also for the identification of distinct types of system goals, such as the distinction between functional (i.e., relating to services that system will provide) and non-functional (i.e., relating to quality characteristics or constraints to which the system must conform) goals [52].

3. Theory of Distributed Cognition

Distributed cognition (DCog) is a branch of cognitive science pioneered by Edwin Hutchins and his colleagues in the 1990s [22, 24, 26, 25, 27, 28]. The central tenet of theory asserts that cognitive processes, such as memory, decision making, and reasoning, are not limited to the mental states of an individual. The development of the theory was motivated by research on teams engaged in complex tasks. In these settings information processing activities are not localized to individuals, but are distributed across members. Furthermore, a significant portion of the cognitive workload is "shouldered" by the technical artifacts employed by group members.

By conceptualizing cognition as "the propagation of representational state across representational media" [24: p. 118], distributed cognition expands the unit of cognitive analysis from that of the individual to that of the entire team attending to a specific task. With this fundamental shift in perspective on cognitive activity, the theory lends itself to at least three significant assertions [25]: 1) the thought process is distributed among members of social groups, 2) cognition employs both internal and external structures, and 3) cognitive processes are distributed over time.

The DCog theory contends that cognitive processes are distributed across members of a group. Each member may play a specific role with respect to the processing of information and the initiation of cognitive action. This idea of *social distribution* has obvious ramifications for the study of distributed RE processes. Nearly all software design efforts are executed through a team structure [18]. Moreover, one essential characteristic of development teams is the diversity of knowledge [8, 36, 57]. While addressing complex design challenges teams must bring together individuals from a wide variety of technical and functional domains. The cognitive task of arriving at stable requirements set, which we referred above as the *computation of requirements*, cannot be localized to any one of these participants, such as a designer (as is often assumed). Rather, it resides in the holistic process of cognitive computation that enables requirements to emerge as a quality of the social system.

The second implication of distributed cognition is that cognitive processes intertwine internal and external structure. While traditional cognitive perspectives focus on the internal states of mind, a DCog approach highlights the ways in which individuals and groups integrate external material elements of the environment as part of their thought processes. The distribution of cognitive activity through the use of external structure is readily apparent in prevailing RE practice. Indeed, the development of formal models in RE can be seen as creating external structures that support subsequent cognitive processes necessary to design. Some of these representations can be materialized in CASE tools that support and integrate cognitive processes embedded in requirements capture and software design [34, 56]. Consequently, existing artifacts serve as a significant *external source* of computing design requirements – setting the initial conditions which both enable and constrain design [2, 63, 19].

Finally, DCog theory contends that cognitive processes may be distributed not only in social and spatial terms, but also with respect to time- i.e. cognition is path dependent. Earlier actions influence the cognitive processes enacted later. Temporal distribution of cognition is present in any context where heuristics have been formulated for generating appropriate cognitive activity. Design efforts draw heavily upon requirements and artifacts inherited from earlier projects. For example, formal information architectures (e.g., enterprise and product architectures) often act as a mechanism to ensure consistency across multiple designs [19]. An extensive literature on requirements reuse suggests multiple approaches to distribute requirements computation over time [9, 35, 44]. In addition, some researchers have investigated temporal distribution of requirements while emphasizing iteration and evolution [e.g., 2, 14, 32].

The three facets of distributed cognition (i.e., social distribution, the use of external structure, and temporal extension) have often been highlighted in isolation. Naturally, they are closely linked in practice: the distribution of cognition over time implies the use of both social transmission (e.g., project team interaction) and material artifacts (i.e., legacy systems, enterprise architecture) to support memory. Similarly, socially-distributed cognitive processes are likely to employ both internal and external structures during individually-intensive cognitive tasks. In all we posit that theory of DCog offers a fruitful lens for assessing the ways in which requirements computation is distributed across individuals, organizations, and artifacts in today's design environments.

4. Research Approach

The present study leverages two complementary research approaches. The initial phase of research was centered on a case study of a complex systems development project. The case analysis provides us the grounding for the identification of development goals that inform a distributed cognitive perspective. In the second phase of the research, we build upon the case to

develop a simulation of goal satisfaction using a GORE/i* modeling tool.

4.1. Case Study Design

As noted DCog processes form an emerging phenomenon that is not subject to straightforward manipulation. Accordingly, a case study approach is warranted to provide an occasion for rich exploration of the practical activities of designers and other stakeholders during RE [13, 58]. Indeed, for this very reason, case studies have been a favored approach for empirical work in RE research [17, 45]. Indeed, several scholars have employed the case studies in their attempts to generate rich theory-yielding insights about RE processes and outcomes [64].

Therefore, we conducted an exploratory case study of RE-related cognition within a development team focusing on the modification and implementation of a large enterprise resource planning (ERP) system at a mid-size university in the Midwestern U.S. The case inquiry was conducted in accordance with prevailing case study field procedures, including the development of a case study protocol prior to data collection, triangulation using multiple sources of evidence, and the maintenance of a chain of evidence [58]. The data collection included interviews, direct observation of project interactions, and documentary review (e.g., specification documents, customization requests, business process models, design mock-ups). Interview transcripts and observational field notes were coded using Atlas.ti. The coding centered on a thematic analysis of the data [6] and was conducted conforming to principles of grounded theory [15, 51]. This included constant comparison and open, axial, and selective coding. Our approach differs from some interpretations of grounded theory in that the final analysis was informed by constructs from RE research, such as goal differentiation, and the DCog theory.

4.2. Case Summary: University SIS Project

In 2006, a mid-sized Midwestern U.S. university initiated the acquisition, customization, and implementation of the PeopleSoft Student Information System (SIS) ERP. The SIS Project was intended to integrate all student information and student-facing administrative functions across the university's nine distinct schools. Key functions supported by the envisioned platform included admissions, financial aid, course selection and enrollment, grading, degree tracking, and transcript management. The initial roll-out of the system was completed in fall 2008, with additional functionality rolled out over the course of the subsequent academic year. The installation of the SIS platform was considered a successful effort, including the management of platform requirements.

The organization is a mid-size private university. The university serves nearly 10,000 students (4,200 undergraduate, 2,200 graduate, and 3,500 professional students) across seven distinct schools. Traditionally, each school managed its own student records, with some aggregation of basic student information in the university's legacy student information system. Different administrative functions were managed using a collection of distinct software applications. The SIS Project was undertaken in an effort to integrate various student-related data sources and functions across the entire university.

The SIS was the third phase of a broader ERP installation program. The university had selected Oracle's PeopleSoft platform as the ERP package. In 2005 and 2006, the university had rolled out two installations of the platform, covering the Financial and Human Capital Management components. The SIS was the final major installation necessary for the achievement of a comprehensive enterprise-level information system serving the university.

4.3. Simulation

In the second phase of the research, we built upon the case analysis findings to analyze the interplay of various goal types and the RE-oriented activities of the SIS project team. Specifically, we created goal-oriented models of the SIS development process using an Eclipse-based i* star modeling tool, jUCMNav [40, 47]. These models incorporated the requirements processes identified in the case analysis as well as the goal taxonomy developed from the initial analysis. The jUCMNav tool was used to create a model of mutual dependencies between requirements tasks and identified goals. The models were then used to conduct a series of simulations to assess the impact of variable execution of requirements tasks on goals at varying levels.

4.4. Goal Model Evaluation

Simulating the evaluation of goal models involves four steps. The first step is to specify goal models from common perspectives [1]. We did so for general project goals, SIS-specific project goal, requirements engineering goals, and distributed cognition goals. A portion of the distributed cognition goals model is shown in Figure 1. (With regard to goal modeling, you may ignore the tasks that are associated with the goals at the bottom of the figures). Each model includes goals and relationships as identified in the literature.

Although incomplete, the models represent the most relevant goals for the problem.

Each oval represents a softgoal, while the links represent contributions that are positive (+) or negative (-) to the satisfaction of the goal at the arrowhead. Goal satisfaction is calculated by propagating values through the goal graph according to the specified node satisfaction value and the weighted contributions [1].

The models include quantitative numbers that indicate the contributions goals have on each other. We chose to model subgoals as totaling 100 percent contribution to allow us to analyze the relative influence of goals and their realized tasks. It implies a complete decomposition. However, our model is incomplete. When a new contributing goal is added, then the contributions are modified to reestablish their aggregation to 100 percent.

The project specific (SIS) model provides a place to add goals that are not common to most projects. Thus, the modeling is a combination of:

1. Creating a project specific (SIS) model for project specific goals.
2. Linking the project specific goals to the existing (pre-defined) goal models
3. Adding project specific tasks and linking them to new project goal model and the existing (pre-defined) goal models

The second step is to specify goal models for the problem. We did so for the SIS project. The model consists of a goal model and a task model. The third step is to specify scenarios for the problem. The variable tasks that we considered are based on observations of the SIS case. These tasks are associated with the goal models. The final step is to evaluate the impact that each scenario has on the goal models. We take this up in the following section on Findings.

5. Findings

Given the two-phase structure of the research effort, we report the findings for each phase separately. As noted above, the design and execution of the simulation phase of the research built upon the findings from the case analysis phase.

5.1. SIS Project RE Activities

The University SIS project supported a number of findings regarding both the nature of the RE tasks pursued and the goals implicit in the processes. The SIS project reflected both higher-level design processes focused on the discovery, specification, and validation of project requirements and lower-level tasks variably employed within the broader RE-oriented processes. At the higher-level, the project employed a four-stage process for progressive elaboration of user requirements. Importantly, our findings revealed that the processes were not executed in a universal manner – i.e., some RE-oriented activities were omitted or bypassed at various times. This variable execution of tasks is relevant for our later simulation of RE outcomes.

Interactive design and prototyping. The initial effort at requirements discovery in the SIS project was called the Interactive Design and Prototyping (IDP) process. The IDP process sought to inform key stakeholders about the functionality of PeopleSoft and to elicit statements of need for customization or modification. Thus, IDP was at its core a gap analysis. The IDP process consisted of JAD-style focus group discussions scheduled with each of the over 100 functional offices on campus. The IDP sessions included the project leadership, functional area leads, and technical experts, and focused on the input of office personnel regarding the appropriateness of the PeopleSoft system for their business functions. The result of each session was the articulation of desired modifications.

Interactive engagement with users. While not formally labeled by the project team, the second core RE task focused on iterative discussion between project functional leads/consultants and user representatives for distinct business units or schools. We have labeled this process Iterative Engagement with Users (IEU). The IEU discussions centered on review of the document developed as part of the IDP process and discussion of specific functional modifications desired by the users. As an outcome of the IEU process, the functional leads/ consultants developed a Preliminary Specification Document and submitted it for review and validation by the users. As the name implies, the IEU process was repeated until users felt that their desired modifications were appropriately captured

Structured walkthrough. Consensus around specifications and change requests on the part of the project team members was achieved through the third RE task, Structured Walkthrough. The walkthroughs were attended by the leadership of the project team, including the Project Director; Functional, Technical, and Project Management Leads; the consulting Project Manager and lead functional and technical consultants; and training team representatives. No users, functional SMEs, or technical experts were in attendance. During the walkthroughs, a specification developer would guide the participants through a detailed discussion of a requested change. Questions were raised and debated by the entire project team. The walkthroughs generally resulted in one of three outcomes: 1) the specification

was accepted and the Technical Lead took responsibility for scheduling modifications, 2) the discussion raised sufficient problems with the current status of the specification so that a decision was made to revise the specification, or 3) the specification was tabled for later discussion.

Design review. The final core RE task employed on the SIS project was the Design Review. In Design Review, a technical developer or consultant met with user representatives to review a proposed resolution. Generally, this task centered on review of solution prototype that the developer had created based on the specification accepted by the project team leadership. If users are satisfied with resolution, then developers would proceed to final implementation of the modification. Conversely, if users desired additional changes to the proposed resolution, then the developer pursued additional prototyping of the solution until satisfaction was achieved. Importantly, of tall the RE tasks outlined, Design Review was the most variable, with the option of prototyping and review left largely to the discretion of individual developers.

Ancillary RE Tasks. In addition to the four high-level RE activities, the SIS project entailed several detail-level RE tasks that were again variably executed over the course of requirements determination. Observed lower-level tasks included the following:

- Business Process Modeling: The development of business process models for distinct schools or individual business units. When executed, the business process modeling was generally associated with the IDP process, and intended to support an understanding of a business unit's current state.
- Scenario Development: The generation of multiple scenarios for design modifications. This task was most commonly observed in the structured walkthrough process, and provided a mechanism for the design team to explore users' stated requirements at a deeper level.
- Mock-ups: The creation of mock-ups or "throw-away" prototypes to illustrate modification options. This rapid prototyping was generally employed as a requirements validation technique and most frequently associated with the Design Review process.

As noted above, all of these RE activities (both higher-level formal processes and detailed tasks) were variably executed on the SIS project. We did not observe any "hard" rules for when a given activities would or would not be executed; rather, the execution of RE tasks appeared to largely reflect individual preferences or design expertise.

5.2. A Goal Taxomony for the SIS Project

In addition to illustrating the different types of RE tasks executed, the analysis of the SIS case revealed the distinct categories of goals that were relevant to the design effort. Specifically, we identified four distinct categories of goals within the SIS project:

- Common project goals: This class of goals represented project objectives that are relevant for almost all IT implementation projects. These goals largely relate to the project management triple constraint of time, cost, and quality/functionality.
- Idiosyncratic project goals: In this class of goals, we identified objectives that appear to be specific to the SIS project or projects of a similar focus.
- RE goals: These are goals associated with commonly-held measures of requirements quality.
- DCog Goals: Perhaps most criticality for the present analysis, we identified a number of goals that are implied by the application of DCog theory. These are characteristics of a cognitive system that will support system effectiveness and robustness, ensuring that the socio-technical system (i.e., people and supporting artifacts) can react to changing conditions and reconfigure its computational structure when necessary (e.g., if a given individual or artifact is removed).

A summary of the resulting goal taxonomy is provided in Table 1.

Table 1. Summary of SIS Goal Taxonomy

Goals	Descriptions
Common Project Goals	
System adoption	Ensuring that users accept and use the functionality provided in the system
Minimize duration	Seeking adherence to project timelines and positive schedule variance
Maximize implemented functionality	Implemented as much system functionality as possible within time and budgetary constraints
Minimize project costs	Managing the project budget to ensure cost effective implementation
Accuracy of status reporting	Keeping executive management informed about the status of the project
Supporting collaboration	Ensuring effective collaborative work among project team members
Idiosyncratic Project Goals	
Minimize platform modifications	Keeping modifications of the platform to the minimum required for desired functional support
Training effectiveness	Ensuring that users were adequately trained for system use
Ensure integration	Achieving data integration between the vendor platform and legacy systems

Minimize process changes	Limiting business process changes to those that were absolutely necessary for effective system use
RE Goals	
Completeness	Ensuring that all substantial requirements are identified and addressed in the design
Consistency	Ensuring that requirements did not conflict with one another
Adequacy	Ensuring that requirements will meet the information needs of stakeholders
Clarity	Avoiding ambiguous requirements (i.e., competing interpretations)
Correctness	Ensuring that stated requirements actually reflect the intent of users
Traceability	Ensuring that requirements can be traced to both relevant business objectives and designed features
DCog Goals	
Maintaining common knowledge	Creating common understanding of system requirements and business process; knowledge redundancy
Clarity of processes	Ensuring that project team members know the processes for requirements identification and incorporation
Transparency of action	Enabling team members to "see" what others members of the system are doing
Common language	Reinforcing shared mechanisms for communicating requirements
Temporal distribution	Embedding requirements knowledge in artifacts; requirements reuse

5.2. Simulation Results

Having established a goal taxonomy for the SIS project, we used the jUCMNav tool to create an i* model of the SIS project. The resulting i* models incorporated the goals identified, their inter-relationship, and their impact on the RE-oriented tasks executed on the project (see Sections 4.4. and 5.1.). Importantly, the tool also enabled us to model the relationships between the goals themselves. While length restrictions prohibit a full presentation of the models generated, Figure 1 presents a portion of the distributed cognitive goals model for illustration.

In addition to modeling the relationships between goals, the jUCMNav tool enabled goal model evaluation based on the four-step process outlined in Section 4.4. For the simulation exercise, we focused on the RE-tasks observed to be most variable in the SIS case: design review, mock-up generation, business process modeling, and individual specification review for structured walkthroughs.

Here we evaluate the impact that each scenario has on the goal models. Table 2 summarizes the values for the root nodes of the three goal models, indicating how much the perspectives are satisfied by each scenario. (Note: The SIS-specific goal model links into the other three models, so the evaluation is reflected in the three other models). Figure 1 illustrates the DCog goal model for one scenario (No Design Review). The value of the goal analysis is in the relative impact that different scenarios have on goal satisfaction.

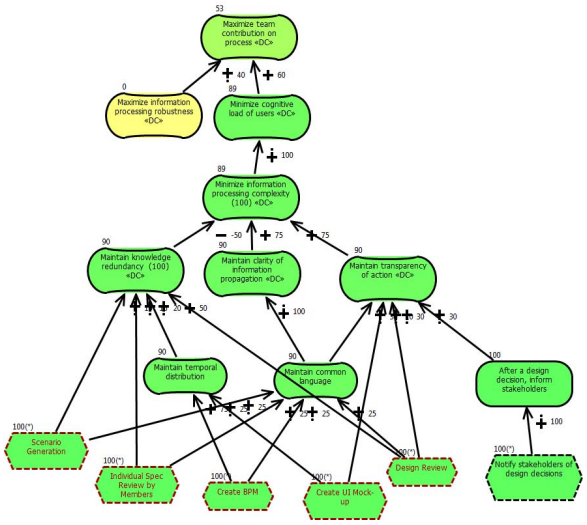

Figure 1. Goal scenario of DCog goal satisfaction derived from SIS task satisfaction

In reviewing Table 2, each row represents a scenario. In the first row, all tasks are included in the development process, while the last row represents no tasks in the development process. The intervening rows show results for the other scenarios evaluated. The scenario "Except Design Review" includes all tasks except the design review task. Notice that its average evaluation of the three root nodes is 58%, which is the lowest average evaluation. Thus, this scenario has the greatest impact on the development process, according to the goal models.

Table 2. Goal satisfaction values of scenarios

Scenarios	System Adoption <<PM>>	Maintain Common Knowl. <<DC>>	Reqs. adequacy <<RE>>	Avg.
All	100%	100%	100%	100%
Except Design Review	46%	75%	52%	58%
Except BPM	100%	91%	88%	93%
Except Ind. Spec. Review	100%	92%	95%	96%
Except Scenarios	100%	92%	95%	96%
Except UI Mockups	100%	98%	97%	98%
No Tasks	0%	0%	0%	0%

6. Discussion and Conclusions

This research presents the initiation of a broader program of study focusing on the role of distributed cognitive processes in the practice of contemporary information systems design. While the study represents a proof-of-concept around the modeling of distributed cognitive dynamics in formal goal models, we believe it suggests several significant contributions to RE research and practice.

First, the study combines the empirical insights of *in situ* case analysis with the simulation and goal evaluation capabilities of i* modeling. In this way, the study illustrates the potential for reorienting RE research from a purely prescriptive outlook to one grounded in the experiences of practicing IS designers.

Secondly, the research extends the theory of distributed cognition through a focus on the practical design principles (i.e., DCog goals) that can be derived from the theory's application as an analytical tool. By applying the theory to an existing IS design context and deriving distinct goals implied by its perspective on socio-technical cognitive systems, we have generated a series of preliminary concepts for subsequent IS development process design and a mechanism for evaluation of their relative efficacy.

Third, the research calls attention to the value of analyzing RE as a socio-technical process which must be approached with an eye to the intricate web of interactions between diverse social actors and the artifacts which they employ. This systems-oriented perspective offers us insights for both addressing persistent challenges to effective RE and capitalizing on opportunities for greater innovation and design breakthroughs.

Finally, the combined case analysis and goal-oriented modeling approach creates a common basis for evaluation of distinct IS development methods. The analysis and modeling process outlined here may be extended to the evaluation of emergent approaches, such as OSSD and agile development. In particular, we are interested in modeling the different computational structures that are implied by these diverse approaches to IS design.

8. References

[1] D. Amyot, "Introduction to the user requirements notation: Learning by example", Computer Networks, 42 (2003), pp. 285-301.

[2] A. Antón and C. Potts, "The use of goals to surface requirements for evolving systems", Proceedings of the 20th international conference on Software engineering (1998), pp. 157-166.

[3] A. Aurum and C. Wohlin, *Requirements Engineering: Setting the Context*, in A. Aurum and C. Wohlin, eds., *Engineering and Managing Software Requirements*, Springer-Verlag, Berlin, Germany, 2005, pp. 1-15.

[4] B. Boehm, "Verifying and validating software requirements and design specifications", IEEE Software, 1 (1984), pp. 75-88.

[5] A. Borgida, S. Greenspan and J. Mylopoulos, "Knowledge Representation as the Basis for Requirements Specifications", IEEE Computer, 18 (1985), pp. 82-91.

[6] R. E. Boyatzis, *Transforming qualitative information: Thematic analysis and code development*, Sage Publications, Inc, Thousand Oaks, CA, 1998.

[7] J. Castro, M. Kolp and J. Mylopoulos, "Towards requirements-driven information systems engineering: the Tropos project", Information Systems, 27 (2002), pp. 365-389.

[8] B. Curtis, H. Krasner and N. Iscoe, "A field study of the software design process for large systems", Communications of the ACM, 31 (1988), pp. 1268-1287.

[9] J. Cybulski, *Patterns in software requirements reuse*, *Third Australian Conference on Requirements Engineering (ACRE'98)*, Geelong, Australia, 1998, pp. 135–153.

[10] A. Dardenne, A. van Lamsweerde and S. Fickas, "Goal-Directed Requirements Acquisition", Science of Computer Programming, 20 (1993), pp. 3-50.

[11] A. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, A. Ta and M. Theofanos, *Identifying and measuring quality in a software requirements specification*, *Proceedings of the First International Software Metrics Symposium*, IEEE Computer Society, Los Alamitos, CA, 1993, pp. 141–152.

[12] M. Dorfman, *Software Requirements Engineering*, in R. H. Thayer and M. Dorfman, eds., *Software Requirements Engineering*, IEEE Computer Society Press, 1997, pp. 7-22.

[13] K. Eisenhardt, "Building Theories from Case Study Research", Academy of Management Review, 14 (1989), pp. 532-550.

[14] N. A. Ernst, J. Mylopoulos and Y. Wang, *Requirements Evolution and What (Research) to Do about It*, in K. J. Lyytinen, P. Loucopoulos, J. Mylopoulos and W. Robinson, eds., *Design Requirements Engineering: A Ten-Year Perspective*, Springer-Verlag, Heidelberg, Germany, 2009, pp. 186-214.

[15] B. G. Glaser and A. L. Strauss, *Discovery of Grounded Theory: Strategies for Qualitative Research*, Aldine Publishing Company, Chicago, IL, 1967.

[16] J. Goguen and C. Linde, "Techniques for Requirements Elicitation", Requirements Engineering, 93 (1993), pp. 152-164.

[17] O. Gotel and A. Finkelstein, *Extended requirements traceability: results of an industrial case study*, *Third IEEE International Symposium on Requirements Engineering*, IEEE Press, Annapolis, MD, USA, 1997, pp. 169-178.

[18] P. J. Guinan, J. G. Cooprider and S. Faraj, "Enabling Software Development Team Performance During Requirements Definition: A Behavioral Versus Technical Approach", Information Systems Research, 9 (1998), pp. 101-125.

[19] S. Hansen, N. Berente and K. J. Lyytinen, *Requirements in the 21st Century: Current Practice & Emerging Trends*, in K. J. Lyytinen, P. Loucopoulos, J. Mylopoulos and W. Robinson, eds., *Design Requirements Engineering: A Ten-Year Perspective*, Springer-Verlag, Heidelberg, Germany, 2009, pp. 44-87.

[20] A. Hickey and A. Davis, "Elicitation technique selection: how do experts do it?", Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International (2003), pp. 169-178.

[21] J. Highsmith and A. Cockburn, "Agile software development: The business of innovation", Computer, 34 (2001), pp. 120-127.

[22] J. Hollan, E. Hutchins and D. Kirsh, "Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research", ACM Transactions on Computer-Human Interaction, 7 (2000), pp. 174-196.

[23] E. Hull, K. Jackson and J. Dick, *Requirements Engineering*, Springer-Verlag, London, UK, 2005.

[24] E. Hutchins, *Cognition in the Wild*, MIT Press, Cambridge, MA, 1995.

[25] E. Hutchins, *Distributed Cognition*, *International Encyclopedia of the Social & Behavioral Sciences*, Elsevier, Ltd., 2000.

[26] E. Hutchins, "How a Cockpit Remembers Its Speed", Cognitive Science, 19 (1995), pp. 265-288.

[27] E. Hutchins, *The technology of team navigation*, in J. Galegher, R. Kraut and C. Egido, eds., *Intellectual teamwork: social and technical bases of collaborative work*, Lawrence Erlbaum Assoc., Hillsdale, NJ, 1990.

[28] E. Hutchins and T. Klausen, *Distributed Cognition in an Airline Cockpit*, in Y. Engestrom and D. Middleton, eds., *Cognition and Communication at Work*, Cambridge University Press, New York, 1996, pp. 15-34.

[29] M. Jackson, *Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices*, ACM Press/Addison-Wesley Publishing Co., New York, NY, 1995.

[30] M. Jarke, J. Bubenko, C. Rolland, A. Sutcliffe and Y. Vassilou, *Theories underlying requirements engineering: An overview of NATURE at Genesis*, *IEEE International Symposium on Requirements Engineering (ISRE'93)*, IEEE Computer Society, San Diego, CA, 1993, pp. 19-31.

[31] M. Jarke, R. Klamma and K. Lyytinen, *Metamodeling*, in M. A. Jeusfeld, M. Jarke and J. Mylopoulos, eds., *Metamodeling for Method Engineering*, MIT Press, Cambridge, MA, 2009, pp. 43-88.

[32] M. Jarke, K. Pohl, S. Jacobs, J. Bubenko, P. Assenova, P. Holm, B. Wangler, C. Rolland, V. Plihon and J. Schmitt, *Requirements engineering: An integrated view of representation, process, and domain*, in I. Sommerville and M. Paul, eds., *Proceedings of the 4th European Software Engineering Conference (ESEC'93)*, Springer-Verlag, Garmisch-Partenkirchen, Germany, 1993, pp. 100-114.

[33] G. Kotonya and I. Sommerville, *Requirements Engineering: Processes and Techniques*, John Wiley & Sons, New York, NY, 1998.

[34] P. Kruchten, *The Rational Unified Process: An Introduction*, Pearson Education, Boston, MA, 2003.

[35] W. Lam, J. A. McDermid and A. J. Vickers, "Ten steps towards systematic requirements reuse", Requirements Engineering, 2 (1997), pp. 102-113.

[36] N. Levina and E. Vaast, "The Emergence of Boundary Spanning Competence in Practice: Implications for Implementation and Use of Information Systems", MIS Quarterly, 29 (2005), pp. 335-363.

[37] M. Loomes and S. Jones, *Requirements engineering: A perspective through theory-building*, *Third International Conference on Requirements Engineering (ICRE'98)*, IEEE Computer Society, Colorado Springs, CO 1998, pp. 100-107.

[38] P. Loucopoulos and V. Karakostas, *System Requirements Engineering*, McGraw-Hill, Inc., New York, NY, 1995.

[39] R. Martin, *Agile Software Development: Principles, Patterns, and Practices*, Prentice Hall PTR Upper Saddle River, NJ, 2003.

[40] G. Mussbacher and D. Amyot, *Goal and scenario modeling, analysis, and transformation with jUCMNav*, *Proceedings of the 31st International Conference on Software Engineering (ICSE'09)* IEEE, Vancouver, BC, 2009, pp. 431-432.

[41] J. Mylopoulos, "Information Modeling in the Time of the Revolution", Information Systems, 23 (1998), pp. 127-155.

[42] J. Mylopoulos, J. Castro and M. Kolp, *Tropos: A framework for requirements-driven software development*, in J. Brinkkemper and A. Solvberg, eds., *Information Systems Engineering: State of the Art and Research Themes, Springer-Verlag*, Springer-Verlag, Berlin, Germany, 2000, pp. 261-273.

[43] J. Mylopoulos, L. Chung and B. Nixon, "Representing and using nonfunctional requirements: A process-oriented approach", IEEE Transactions on Software Engineering, 18 (1992), pp. 483-497.

[44] B. Nuseibeh and S. Easterbrook, *Requirements Engineering: A Roadmap*, *Proceedings of the conference on The future of Software engineering*, ACM Press New York, NY, USA, Limerick, Ireland, 2000, pp. 35-46.

[45] B. Ramesh and M. Jarke, "Toward reference models for requirements traceability", IEEE Transactions on Software Engineering, 27 (2001), pp. 58-93.

[46] W. N. Robinson, "Integrating multiple specifications using domain goals", ACM SIGSOFT Software Engineering Notes, 14 (1989), pp. 219-226.

[47] J. F. Roy, J. Kealey and D. Amyot, *Towards integrated tool support for the User Requirements Notation*, in R. Gotzhein and R. Reed, eds., *System Analysis and Modeling: Language Profiles*, Springer-Verlag, Berlin, DE, 2006, pp. 198-215.

[48] W. Scacchi, "Free/open source software development: Recent research results and methods", Advances in Computers, 69 (2007), pp. 243-295.

[49] I. Sommerville, "Systems engineering for software engineers", Annals of Software Engineering, 6 (1998), pp. 111-129.

[50] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*, John Wiley & Sons, Inc. New York, NY, USA, 1997.

[51] A. L. Strauss and J. Corbin, *Basics of qualitative research: Grounded theory procedures and techniques*, Sage, Newbury Park, CA, 1990.

[52] A. van Lamsweerde, *Goal-Oriented Requirements Engineering: A Guided Tour*, *Proceedings of the 5th IEEE International Symposium on Requirements Engineering*, IEEE, Toronto, Ontario, 2001, pp. 249-263.

[53] A. van Lamsweerde, "Requirements Engineering in the Year 00: A Research Perspective", Proceedings of the 22nd international conference on Software engineering (2000), pp. 5-19.

[54] A. van Lamsweerde, R. Darimont and E. Letier, "Managing conflicts in goal-driven requirements engineering: Managing inconsistency in software development", IEEE Transactions on Software Engineering, 24 (1998), pp. 908-926.

[55] I. Vessey and S. A. Conger, "Requirements Specification: Learning Object, Process, and Data Methodologies", Communications of the ACM, 37 (1994), pp. 102-113.

[56] I. Vessey and A. P. Sravanapudi, "CASE tools as collaborative support technologies", Communications of the ACM, 38 (1995), pp. 83-95.

[57] D. B. Walz, J. J. Elam and B. Curtis, "Inside a Software Design Team: Knowledge Acquisition, Sharing, and Integration", Communications of the ACM, 36 (1993), pp. 63-77.

[58] R. K. Yin, *Case Study Research: Design and Methods*, SAGE Publications, Thousand Oaks, CA, 2009.

[59] E. Yu, *Towards modelling and reasoning support for early-phase requirements engineering*, *Proceedings of the Third IEEE International Symposium on Requirements Engineering (ISRE'97)*, IEEE Computer Society, Annapolis, MD, 1997, pp. 226-235.

[60] E. Yu and J. Mylopoulos, *Why Goal-Oriented Requirements Engineering*, *Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality*, Pisa, Italy, 1998, pp. 15-22.

[61] Y. Yu, J. Leite and J. Mylopoulos, *From goals to aspects: Discovering aspects from requirements goal models*, *Proceedings of the 12th IEEE International Requirements Engineering Conference (RE'04)*, Kyoto, Japan, 2004, pp. 38-47.

[62] K. Yue, *What does it mean to say that a specification is complete?*, *Proceedings of the Fourth International Workshop on Software Specification and Design (IWSSD-4)*, Monterey CA, 1987.

[63] J. Zachman, "Enterprise Architecture: The Issue of the Century", Database Programming and Design, 10 (1997), pp. 44-53.

[64] P. Zave, "Classification of research efforts in requirements engineering", ACM Computing Surveys, 29 (1997), pp. 315-321.