Contents lists available at ScienceDirect

Interacting with Computers

journal homepage: www.elsevier.com/locate/intcom

An integrative approach to requirements analysis: How task models support requirements reuse in a user-centric design framework

Cyril Montabert [a], D. Scott McCrickard [a,*], Woodrow W. Winchester [b], Manuel A. Pérez-Quiñones [a]

[a] *Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA*
[b] *Grado Department of Industrial and System Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA*

A R T I C L E   I N F O

Article history:
Received 3 January 2007
Received in revised form 5 June 2009
Accepted 7 June 2009
Available online 10 June 2009

Keywords:
Requirements engineering
Task modeling
Reuse
Critical parameters

A B S T R A C T

Many software systems fail to address their intended purpose because of a lack of user involvement and requirements deficiencies. This paper discusses the elaboration of a requirements-analysis process that integrates a critical-parameter-based approach to task modeling within a user-centric design framework. On one hand, adapting task models to capture requirements bridges the gap between scenarios and critical parameters which benefits design from the standpoint of user involvement and accurate requirements. On the other hand, using task models as a reusable component leverages requirements reuse which benefits design by increasing quality while simultaneously reducing development costs and time-to-market. First, we present the establishment of both a user-centric and reuse-centric requirements process along with its implementation within an integrated design tool suite. Secondly, we report the design, procedures, and findings of two user studies aimed at assessing the feasibility for novice designers to conduct the process as well as evaluating the resulting benefits upon requirements-analysis deliverables, requirements quality, and requirements reuse.

1. Introduction

The ability to maximize the chances for a software project to succeed is an ongoing challenge in every sector of the industry. Despite the establishment of a countless number of development processes and best practices, over 50% of projects are reported as challenged due to significant cost overruns, major delays, and deliverance with only partial functionalities, while more than 30% simply have to be aborted (The Standish Group, 1994). These challenged and terminated projects cost companies billions of dollars every year just in wasted development efforts. Requirements inconsistencies and lack of user involvement have been identified as the principal cause in project failure (European Software Institute, 1996; The Standish Group, 1994; Thayer and Dorfman, 1997). On one hand, the requirements-analysis phase is crucial for the design outcome of any software system since a software solution is successful only to the extent where it meets its intended purpose. Overlooking the requirements phase or establishing requirements that fail to properly reflect the real needs of users is frequent (Berry et al., 2005; Holtzblatt and Beyer, 1995; Martin, 1984) and can be extremely costly to rectify, as a late correction of requirements deficiencies can impact the costs by a factor of 200 (Boehm, 1981). Although it is necessary for requirements specifications to capture the nature of users' activity and equally important for these requirements to be accurately established early in the development process, obtaining good requirements is a difficult enterprise that can only be achieved through a methodical process (Bell and Thayer, 1997; Berry et al., 2005; Brooks, 1987; Wiegers, 2003).

It seems therefore both important and necessary to provide designers with an adequate requirements-engineering infrastructure that facilitates the elicitation of quality requirements. However, user involvement is also a vital factor for project success. Designers can gain significant knowledge from user involvement, an understanding of not only users but also of the work practices and the context within which activities are undertaken. This resulting in-depth understanding of user activities can yield the capture of requirements and thus contribute to the achievement of more effective design solutions (Beyer and Holtzblatt, 1999; Gould and Lewis, 1985; Kujala et al., 2005). Although the benefits resulting from a strengthening of the role of users upon project qualities can hardly be debated, the economical constraint associated with most endeavors and the reluctance from the managerial corpus to reallocate human resources can hinder user involvement tremendously (Mirel, 2000; Wilson et al., 1997). As such, it is therefore critical that a platform for requirements analysis capable of both offering user centricity and fostering requirements quality at minimal expenditure be established.

* Corresponding author. Tel.: +1 540 231 6698; fax: +1 540 231 6075.
E-mail address: mccricks@cs.vt.edu (D. Scott McCrickard).

Because we feel that requirements veracity is the cornerstone for software project success—delivery within allocated timeframe and budget of a system that supports clients' real needs—and because we highlight a strong correlation between user involvement, requirements, and costs, to safeguarding project success, this paper discusses the foundation of a requirements-analysis process that addresses the key attributes of user centricity and quality requirements and outlines its implementation within an integrated design tool suite. We begin by describing the need for formal design approaches and metrics to capture and leverage key elements of a planned design, focusing on critical parameters as a means to define a domain creatively and scientifically, on scenarios and scenario-based design as a means to elaborate and contextualize the problems and planned solutions in a design, and on task analysis as an approach that explicitly represent the actions and processes that users need to undertake when conducting an activity. We then present the design, procedure, and results of two user studies aimed at assessing both the feasibility and the payoff resulting from the deployment of our proposed requirements methodology. Finally, we present our conclusions and suggest future research direction for this line of work.

2. Background and related work

To address the critical issues of lack of user involvement and quality requirements, we first need to investigate requirements-analysis techniques and their potential relationship to one another as it has been argued that "technique integration provides the best avenue for improving requirements engineering" (Sutcliffe, 1997).

Critical parameters are a set of formal metrics or attributes that allows designers to assess whether a system serves its purpose (Newman, 1997). These modeling parameters are judged critical because the success or failure of software projects lay within the degree to which the targeted parameter values are reached. These figures of merit present the characteristics of being persistent for a given class of design problems while also being widely accepted. The invariant nature of critical parameters for a particular class of design problems offers the ground for a lingua franca that could enable designers to set up a framework for identifying, relating, and comparing classes of design problems within a given problem domain. In addition, once critical parameters have been identified, they do not need to be reestablished each time a system is developed—which makes it worthwhile to develop tools upon them. Because critical parameters are inherently vital to the success of a system, the line of focus they offer can greatly simplify the reach for requirements quality. In fact, because critical parameters are manipulable they can provide the foundation for the establishment of predictive models that help promote requirements validity and reduce the need for prototyping (Newman et al., 2000). Although critical parameters alone do not have the potential to be regarded as a driving entity for design, they provide a foundation for setting up design objectives (Newman, 1997). Establishing desirable critical-parameter levels systematically as a part of the requirements-analysis phase can dramatically increase the accuracy of requirements and thus hinge whether a project will succeed or fail.

At the requirements phase, the adoption of user-centric approaches such as *scenario-based design* may help address the dire need for user involvement. Through the elaboration of scenario of use, designers identify and highlight users' experiences, goals, and needs with respect to the technology (Carroll and Rosson, 1992; Rosson and Carroll, 2002). Despite the widespread use and popularity of the technique, scenario-based design remains criticized for its recurrent bias, potential vagueness, and impending lack of coverage (Diaper, 2002). Because of the criticality of quality requirements in project success, these drawbacks rule out a stand-alone reliance on scenario-based requirement analysis. Furthermore, scenario-based design does not offer sufficient built-in and explicit techniques needed for capturing the critical-parameter requirements of a system. Because success depends on both user involvement and proper requirements, there is a crucial need for the establishment of a technique capable of bridging the gap between scenarios and critical parameters.

A successful approach for formalizing and leveraging designers' understanding of users' activities is *task analysis* (Taylor, 1991). While task analysis can take multiple forms, the core objective of this procedure remains the characterization and representation of the physical actions and/or mental processes that users need to undertake when conducting an activity. When conducting a hierarchical task analysis (Annett, 2003; Annett and Duncan, 1967), designers study and decompose the required tasks users perform while carrying out an activity. The tasks constituting the activities are systematically broken down into subtasks which are more manageable, until the desired level of granularity is attained. The resulting work-product of a hierarchical task analysis process is a *task model*, a graphical representation usually in the form of a tree graph that explicitly enumerates all the tasks constituting an activity as well as the hierarchical and temporal relationship between these tasks. On one hand, the task centricity of hierarchical task models makes the approach a candidate for the characterization of these tasks in terms of critical parameters. On the other hand, this formalization of tasks can leverage the accuracy of the description of users' activities (Paternò et al., 1997), contribute to the disambiguation of user activity while allowing for the identification of inconsistencies, and reinforce designers' understanding of a usage situation, making task models a solid infrastructure for requirements analysis (Souchon et al., 2002) capable of mitigating scenario downsides. An integrative approach to requirements analysis making use of critical parameters and task models within a scenario-based framework (Montabert and McCrickard, 2006) shows potential to address both the structure and formalism required for an accurate depiction of systems requirements (Richardson et al., 1998) as well as foster user involvement—key attributes in project success.

An influential factor for requirements quality at minimal expenditure is reuse; specifically, the reuse of design components that provide insight and direction for the end product. Constructing with components that have been tested, verified, and validated not only reduces costs but can also fuel productivity and make for better products (Grady, 1997; Lim, 1994; Matsumoto, 1982; Matsumoto, 1993; Sutcliffe, 2000). In the software world, where less than 15% of novelty is introduced into a new project, a successful implementation of systematic reuse mechanisms would tremendously impact costs and quality (Zand and Samadzadeh, 1994). While efforts to integrate reuse within the software community have been made (Estublier and Vega, 2005), reuse is often introduced too late in the development process. Applied to the field of HCI, reuse strategies largely have been limited to the superficial reutilization of interface components. To capitalize on the benefits of reuse, Sutcliffe (2000) advocates the introduction of reuse as early as possible in the development cycle. Despite the fact that the software community unanimously acknowledges the importance of early reuse within the software development process, few efforts have been spent establishing requirements reuse and little progress has been made (van Lamsweerde, 2000). This paper particularly focuses on the ways that our *design-by-reuse* approach, in which designers leverage existing task models through identification of critical-parameter values, can lead to improved design requirements.

Issued from the detailed analysis of scenarios within a scenario-based design process, claims encompass the key issues contained in scenarios and can serve as a foundation for the capture of reusable design knowledge (Sutcliffe, 2000; Sutcliffe and Carroll, 1999). Previous research efforts have shown the effectiveness of a critical-parameter-based taxonomy for the indexing of knowledge reuse (Fabian et al., 2004). However, because of the abstraction gap between critical parameters and scenarios, a sole reliance on scenario-based design could fail to provide designers with a mechanism that allows the establishment of the desirable critical-parameter levels, thus restraining the true reuse potential of claims at the requirements phase. The task centricity and hierarchic nature of task models can facilitate the acquisition of the classification levels, yielding a significant step toward the establishment of an effective reuse solution (Krueger, 1992).

The domain of *notification systems*—the realm of secondary agents immerged into attention-divided environments (e.g. Ishii and Ullmer, 1997; van Dantzich et al., 2002)—presents valuable attributes for the purpose of our research work. In particular, three clearly identified and recognized critical parameters—interruption, reaction, and comprehension (*IRC*)—reflect desired attributes of a notification system that must be taken into account during requirements gathering and design (McCrickard et al., 2003). Any notification system, existing or planned, has an *IRC* value associated with it: each of *I*, *R*, and *C* ranges from 0 to 1, with a value of 0 indicating that the parameter is not desired by users of the system and a value of 1 indicating that it is. It is possible for the values to be any number between 0 and 1, but prior work suggests the most interesting and important systems tend to have values of or close to 0 or 1 (McCrickard et al., 2003).

The different possible values for the critical parameters define the design space for notification systems. For example, an *IRC* value of 1 1 0 (where *I* is 1, *R* is 1, and *C* is 0) is described as an *alarm* that is intended to interrupt its users with information requiring an immediate reaction; such as a calendar reminder of an upcoming appointment. Conversely, an *IRC* value of 0 0 1 is described as *ambient media* that is intended to raise the comprehension level of its user with minimal interruption or external reaction; such as a constantly updated display of temperature that resides on the computer desktop. Each of the eight possible *IRC* values reflects a different sub-category of systems within the notification systems domain, as detailed in (McCrickard and Chewar, 2003; McCrickard et al., 2003).

Important in the establishment of *IRC* levels are consistent and accurate methods for calculating critical parameters. For existing systems, *IRC* values can be determined by conducting usability tests to determine core usability components like cost of interruption, relative response time, and hit rate, then using those components to calculate *IRC* values (Chewar et al., 2004a,b). Determining *IRC* values for planned systems is more speculative, requiring designers to assess the needs of the users toward identifying the system goals. Yet without consistency among designers on a design team in the determination of critical parameters, the usefulness of the system would be severely limited. One approach to ensure consistency is through a series of multiple choice questions for designers that requires them to speculate on four key concerns: the desired effects (user goals) of receiving a notification, the relative importance and relationship between tasks, the relationship between the notification and existing knowledge, and the monitoring and interpretation effort required due to characteristics of the notification. Tool development and testing reflected that the overall accuracy of *IRC* resulted in a difference of less than 20% for each of the three values (Chewar et al., 2004a,b). This tool, and, more importantly, the characteristics of the *IRC* framework, were essential in the work in this paper.

The *IRC* value reflects that user attention is a limited resource within the notification systems domain that must be redirected from the current activity to gain crucial information. As such, the success of this genre of systems depends significantly on their ability to exhibit the adequate critical-parameter levels. The field offers auxiliary development infrastructures such as a claims library (Fabian et al., 2004; Payne et al., 2003) relying on critical parameters, generalized tasks, and generic tasks nomenclature (Sutcliffe, 2002) as well as an integrated design tool suite—LINK-UP—aimed at providing guidance to the design of notification systems (Chewar et al., 2004a,b). To capitalize on these previous initiatives, we focus the implementation and evaluation of our integrative approach to requirements analysis on this particular class of systems within the LINK-UP framework.

3. An integrative approach to engineering requirements

To address the attributes of user involvement, requirements quality, and low costs essential to project success, we support the integration of alternative yet complimentary techniques for engineering requirements that can potentially bring the best of all worlds. The methodology we are proposing integrates a critical-parameter-based approach to task modeling within a user-centric design framework.

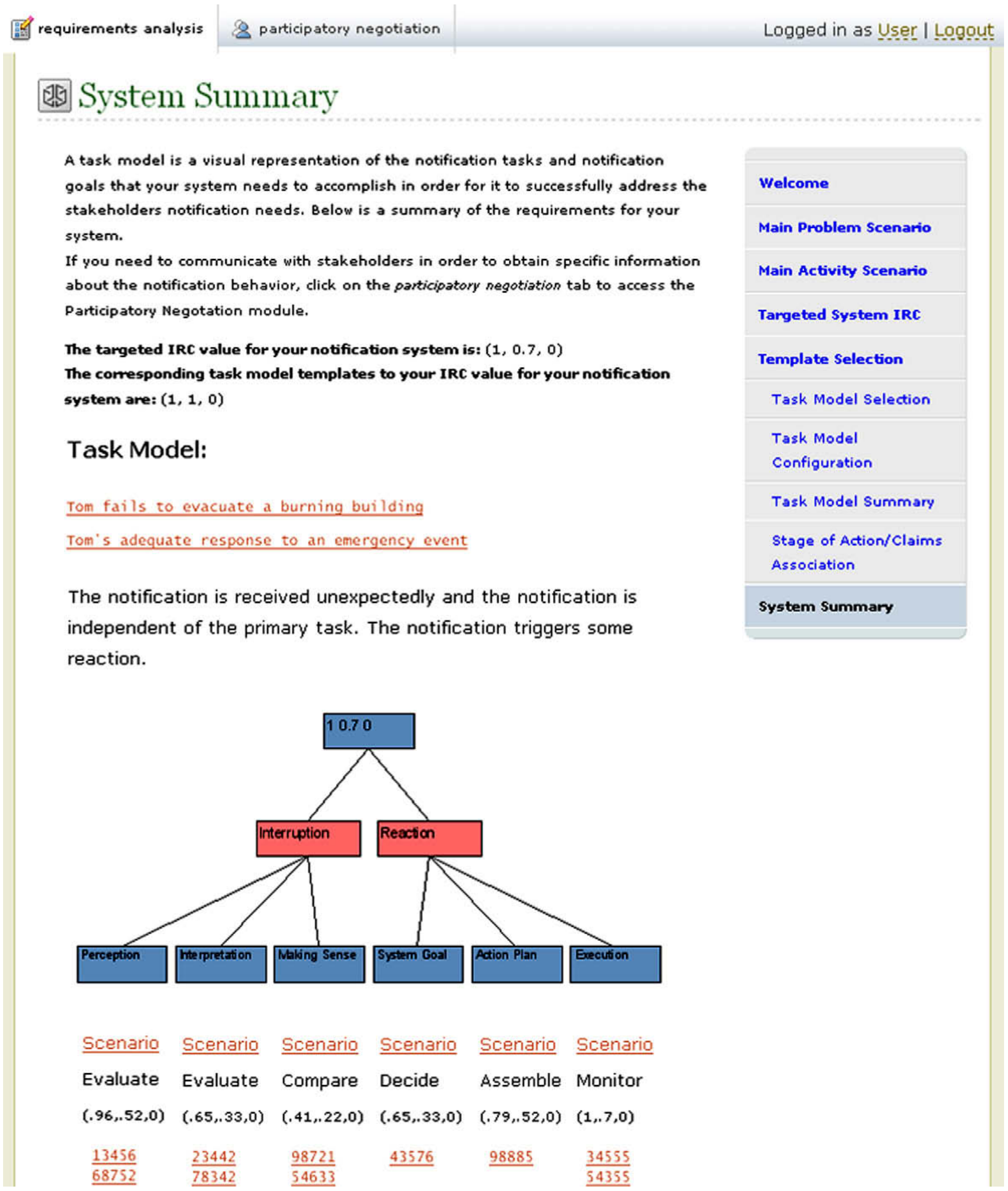
3.1. Critical-parameter-based task models for requirements capture

To assist designers with the formulation of desirable critical-parameter levels and to bridge the abstraction gap between scenarios and critical parameters, we presented a task-modeling approach centered on critical parameters (Montabert et al., 2005) which involves a systematic hierarchic decomposition of tasks into their stages-of-action constituents along with a simultaneous decomposition of their critical-parameter characterizations. In fact, because a high-level task encompasses a user interaction, it can be decomposed in terms of stages of action (Norman, 1986). The stages of action organize subtasks involved when users form cognitive relationships as they interact, whether through physical actions and/or mental processes with a system. The stages of perception, interaction, and making sense allow users to understand the information the system communicates while the stages of forming a goal, establishing an action plan, and carrying its execution, allow users to act upon the system to achieve their desired state. Because Sutcliffe's generic tasks pattern the simple unit procedures that are carried out to accomplish a single goal and exhibit a granularity level similar to the stages of Norman's model of action, they can subsequently be used to characterize the cognitive and/or physical activity that occurs at each stage (Sutcliffe, 2002). Furthermore, although critical parameters numerically capture the criteria for success of an interaction, such levels may not be persistent throughout the crossing of each of the stages of action that define an activity but may rather translate into a particular sequence of critical-parameter objectives. We consequently motivate for the recognition of an additional level of critical parameters that can be used to numerically characterize not the objectives of an entire task that we refer to as task-level critical-parameter specifications but the objectives of an individual stage of action that we refer to as *stage-of-action-level critical-parameter specifications* (SOA-level critical-parameter specifications). Designers can then associate claims to characterize the key features associated with each stage of the activity. Not only does such an approach to task modeling extend the traditional hierarchical task analysis and offer structure during the requirements phase, but each task model also creates a more comprehensive and formalized embodiment of a task's requirements.

In the domain of notification systems for instance, the critical-parameter-based task-model notation presents the class corresponding to the high-level critical-parameter specifications

(McCrickard et al., 2003) at the root of the tree. The *IRC* constituents are then decomposed and related to the stages of action they coerce. Finally, generic tasks are associated to the stages of action to describe the basic stage-based tasks, then characterized by SOA-level critical-parameter specifications.

Fig. 1 describes an example critical-parameter-based task model. As a first step in designing a notification system, a design team would identify one (or perhaps more) target types of notification systems from the eight possible types—in the case of this example, an alarm. This identification of types takes place through selection of critical-parameter values—again for this example, by answering a series of multiple choice questions that highlight the importance of the three critical parameters of interruption, reaction, and comprehension. Each type of notification system has at least one associated task model that breaks down the critical parameter classification into an ordering of the parameters (e.g. interruption followed by reaction) and then an assignment of stages of action to critical parameter (e.g. perception–interpretation-making sense to interruption). Designers can then perform SOA-level critical-parameter specifications. We hypothesize that the discussions and debates that occur within the design team lead to a deeper understanding of the task at hand and a better set of requirements; a hypothesis explored in Section 5.



System Summary

A task model is a visual representation of the notification tasks and notification goals that your system needs to accomplish in order for it to successfully address the stakeholders notification needs. Below is a summary of the requirements for your system.
If you need to communicate with stakeholders in order to obtain specific information about the notification behavior, click on the *participatory negotiation* tab to access the Participatory Negotation module.

The targeted IRC value for your notification system is: (1, 0.7, 0)
The corresponding task model templates to your IRC value for your notification system are: (1, 1, 0)

Task Model:

Tom fails to evacuate a burning building

Tom's adequate response to an emergency event

The notification is received unexpectedly and the notification is independent of the primary task. The notification triggers some reaction.

Scenario	Scenario	Scenario	Scenario	Scenario	Scenario
Evaluate	Evaluate	Compare	Decide	Assemble	Monitor
(.96,.52,0)	(.65,.33,0)	(.41,.22,0)	(.65,.33,0)	(.79,.52,0)	(1,.7,0)
13456	23442	98721	43576	98885	34555
68752	78342	54633			54355

Fig. 1. An example critical-parameter-based task model for a notification task exhibiting an alarm behavior. The critical-parameter constituents of the alarm behavior are decomposed (interruption and reaction) and related to their governing stages of action. Sutcliffe's generic tasks are used to describe the basic activity of each stage. SOA-level critical-parameter specifications capture the objective of each corresponding stage on a 0-to-1 scale as is identified during design processes.

3.2. Critical-parameter-based task models as a reuse catalyst

Although we explored in Fabian et al. (2004) the critical-parameter-based classification method for claims within a knowledge repository, we also unveiled some of the difficulties associated with the acquirement of such attributes. A reliance on critical-parameter-based task models can be a major advance in addressing the criteria selection (Krueger, 1992), a vital facet for the setup of a successful reuse solution at the requirements phase. As a reuse mediator, designers can utilize the artifacts issued from the systematic hierarchical decomposition involved with the critical-parameter-based task-modeling approach to extract relevant knowledge from the repository. In fact, by using the subtasks which describe the nature of the activity at the stages of action along with the SOA-level critical-parameter values which encompass their objective, designers have access to search attributes offering a finer level of granularity to characterize the lowest level of the hierarchic model out of reused components, while the structure of the modeling process leverages simultaneously the veracity associated with the capture of these attributes.

Furthermore, because tasks constitute a shared body of knowledge (Whittaker et al. 2000), we can capitalize on the task centricity of critical-parameter-based task models to leverage reuse further. In fact, since task models detail the important tasks of a system and tasks are often similar across many projects, it is possible for task models to be generalized and reused—particularly if the tasks are sufficiently generalized (Lim and Long, 1994; Wurdel et al., 2007). Critical parameters provide a means to generalize tasks within a broader domain, and the identification of critical parameters for a domain (in the case of our work, the notification systems domain) and the calculation of values for the parameters that match the design goals facilitate the identification of task-model templates. Creating generic task-model templates along with related basic tasks can provide designers with a standard and reusable starting point for requirements analysis that can be employed in new situations, providing support for the design-by-reuse paradigm already implemented through the reuse of claims.

Finally, it is also imperative for the effective implementation of a reuse mechanism to address the design-for-reuse paradigm. Because software projects often only entail a minimal amount of novelty (Zand and Samadzadeh, 1994), requirements associated with a project constructed from an instantiated task model should be relevant to other projects sharing a similar instantiated task model, which thus exhibits a tremendous reuse potential. By preserving, indexing, and promoting the reutilization of critical-parameter-based task models instantiated from generic templates, we facilitate providing designers with a ready-at-hand platform capable of yielding the reuse of claims sets, rationales, and entire project constituents (Montabert and McCrickard, 2006)—a reuse solution at the requirements phase.

3.3. Reuse-centric and user-centric requirements analysis

The requirements-analysis process that we are proposing integrates task modeling and scenario-based design to support and promote the systematic capture of the desirable critical-parameter requirements of an interactive system—figures of merit that have the potential to pave the way to design quality and increase chances of success for the resulting artifact. Key attributes to this approach to requirements analysis are a strong reliance on user involvement, disambiguation and validation of requirements through structure and formalism, and extensive knowledge reuse. First, we anticipate that strong user involvement at the requirements phase may yield the capture of a true reflection of actual needs and expectations as well as facilitate the acceptance of the hypothetical system once completed (Beyer and Holtzblatt, 1999; Mirel, 2000; Wilson et al., 1997). Second, through the introduction of structure and formalism via task modeling, we expect to leverage an in-depth understanding of both the activities and the environment within which these activities are evolving, which will result in the elicitation of true and accurate requirements (Richardson et al., 1998). Third, we foresee an extensive reliance on knowledge reuse at the requirements phase will maximize the reuse payoff and improve the quality of the formulated requirements while simultaneously reducing project costs and shortening time-to-market (Sutcliffe, 2000). Although uniting scenarios, task models, and critical parameters promise tremendous inference for project success, how should this integrative approach to requirements engineering be structured to concretize such potential payoff?

To structure this reuse-centric and user-centric requirements-analysis process, we recommend commencing according to the basic steps and activities recommended by scenario-based design best practice (Carroll and Rosson, 1992; Rosson and Carroll, 2002). The initial step involves an investigation of current work practices. A formulation of a root concept allows setting up the high-level goals for the project and ensures stakeholders share a common initial vision. This high-level vision can then be refined through conducting various field studies, interviews, and artifact investigations as well as other ethnographic inquiries. Problem scenario elicitation ensues, an accurate description of users and their activities in the problem domain, a phase throughout which stakeholders can intervene and bring revisions and validations. Once the problem domain has been adequately framed in a problem scenario, the activity phase follows. This second phase of the requirements-analysis process focuses on the services the hypothetical system needs to support to address the situation crafted in the problem scenario. These services are accurately narrated within an activity scenario. On one hand, because we are putting forth a process centric to knowledge reuse in the form of claims relying on a critical-parameter-based taxonomy and identified an abstraction gap between scenarios and desirable critical-parameter specifications, we need to setup a mediation avenue prior to the conduct of the claims analysis. On the other hand, because the activities described within activity scenarios belongs to the analytic domain, user involvement alone fails to guarantee proper validation. Since the services described in scenarios involve tasks, it is possible to formalize these tasks through a hierarchical task analysis. In fact, Paternò and Mancini (1999) motivate that in order to achieve requirements veracity, one should start with an informal scenario and extract information from it to create and formalize the task model. It then becomes possible to characterize each of the tasks involved in the design model (Norman, 1986) in terms of critical parameters and decompose these task-level critical parameters further through the reuse or the elaboration of critical-parameter-based task models. Coupled with a knowledge repository, these critical-parameter-based task models can serve as a foundation for effective requirements knowledge transfer to occur in the form of claims in lieu of traditional scenario-based claims analysis (Carroll and Rosson, 1992; Rosson and Carroll, 2002).

We anticipate the formalism and structure involved with the constitution of the task models to leverage an in-depth understanding of the activity, improve scenario coverage, allow for the identification of inconsistencies or misspecifications, and ultimately contribute to the elicitation of true requirements, while the reliance on the task models themselves offers refined characterizations of the activities, which also contribute to accurate requirements. Next, we discuss the key implementation facets and rationale of this integrative requirements-analysis methodology within the requirements tool of the LINK-UP system (Chewar et al., 2004a,b).

4. A reuse-centric and user-centric infrastructure for engineering requirements

Although we have established the overall structure of our integrative requirements-engineering process, we still need to identify and organize the steps involved in the conduct of each of the entailed activities. Fig. 2 presents a flowchart of the key activities supported by the infrastructure along with the relationship between each major phase and the knowledge repository. To support verification and refinement, the tool implements the requirements process in an iterative fashion. Furthermore, to promote user involvement beyond early system analysis, during each activity the requirements tool enables designers to bond directly with the negotiation tool of the LINK-UP system (Chewar et al., 2004a,b). This ready-at-hand accessibility to the external module enables designers to engage stakeholders into participatory negotiation sessions where ideas can be exchanged, refined, and validated.

After the completion of the preliminary examinations involved in the elaboration of the root concept and various studies of the work practices in their natural settings, designers are ready to access the requirements tool. Initially, an introductory page provides a comprehensive description of the objectives of requirements analysis, an overview of the supported key activities, and emphasizes the decisiveness of such phase with respect to project success. The rationale for this page is to sensitize designers to the importance of requirements engineering and motivate them to conduct this phase adequately.

4.1. Scenario-based domain analysis

To support the basic steps of a scenario-based requirements analysis, the proposed tool supports the elaboration of a problem and an activity scenario. First, the key activities, supporting artifacts, and social context of the workplace within which activities take place identified during the preliminary examinations are crafted in a problem scenario. The elaboration of such scenario description enables designers to formulate aspects of the stakeholders and their activities in the problem domain that have implications for design. After having clarified the scope and objective of problem scenarios, the requirements tool allows designers to provide a scenario title which should encompass actors, plot, and

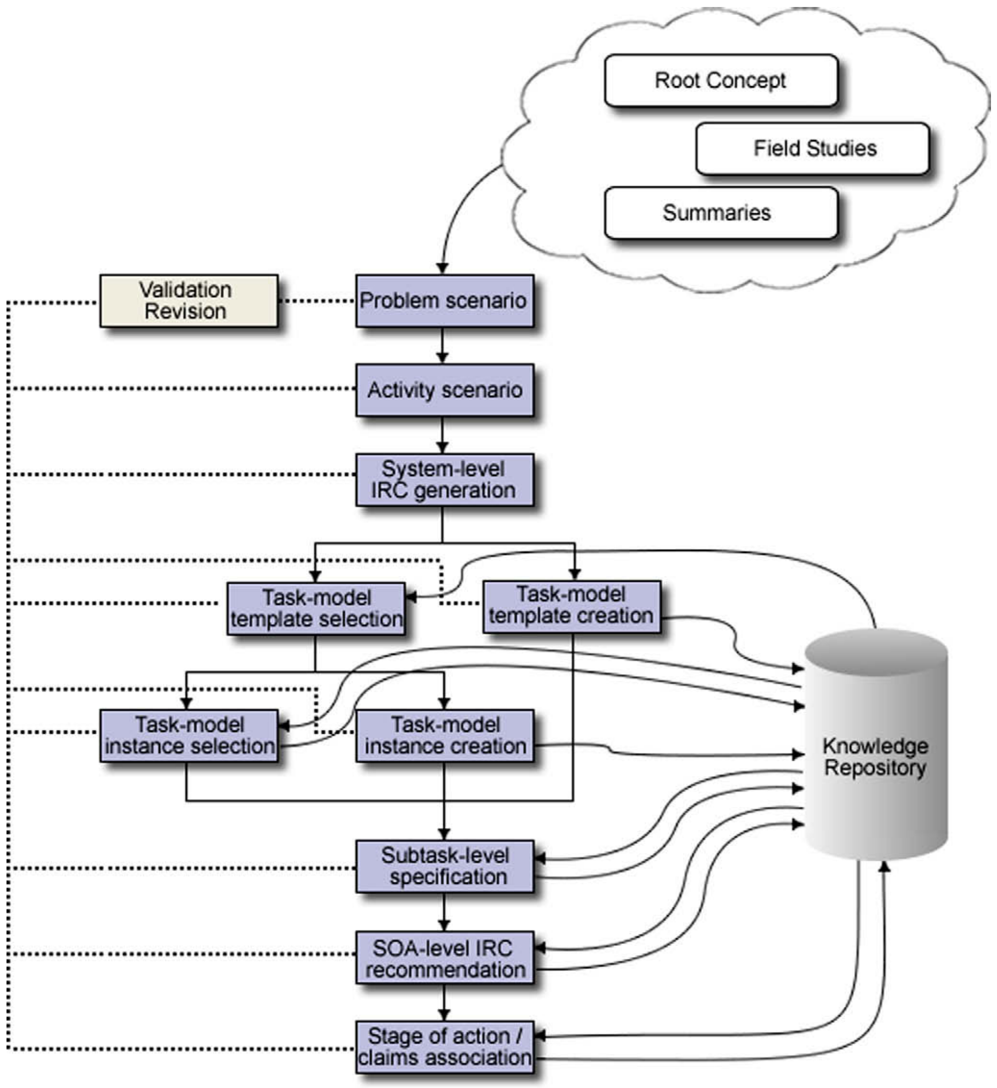

Root Concept
Field Studies
Summaries
Validation Revision
Problem scenario
Activity scenario
System-level IRC generation
Task-model template selection
Task-model template creation
Task-model instance selection
Task-model instance creation
Knowledge Repository
Subtask-level specification
SOA-level IRC recommendation
Stage of action / claims association

Fig. 2. Flowchart of the supported activities with an integrative view of Carroll's scenario-based requirements-analysis process as well as their relationship with the knowledge repository. The stages connected by straight lines are adopted from the early stages of scenario-based design, adapted to account for key elements of critical-parameter generation and task modeling. Curved lines indicate stages where design knowledge is taken from and added to the knowledge repository. Dotted lines indicate the iterative nature of the activities.

scope before they can enter their narration. Following the formulation of the problem scenario, users and their interaction with an envisioned system are described within an activity scenario. The elicitation of such narration allows designers to recognize and mine the activities the target system needs to support to address users' real needs. After having clarified the scope and objective of activity scenarios, the tool allows designers once again to specify a scenario title exhibiting similar attributes as the problem scenario stage prior to the entering of the scenario description. Encouraging designers to specify such peculiar traits within the scenario titles enables users reviewing the coverage and adequacy of the description to quickly identify the designers' discernment of the key elements of the narration. By following such scenario-based domain analysis, the tool ensures an early user involvement and a user-focused base within the requirements phase.

4.2. Task-modeling activity

Once the validation criteria for the scenario phase have been satisfied, we can turn to a task-centric formalization of the activity scenario, as suggested by Paternò and Mancini (1999). In fact, Lafrenière (1996) reinforces the complementarities of both approaches by arguing scenarios should be regarded as a concrete user-oriented approach to depict people's tasks, thoughts, and aspirations, whereas hierarchic task models represent an abstract system-engineer oriented approach that formally describes tasks in a context independent form. However, because our requirements tool is grounded onto the domain of notification systems and we assume that this class of systems supports a single notification task, it is not necessary for the tool to sustain a traditional hierarchical task analysis infrastructure. Rather, the tool directly bridges toward the formulation of the critical-parameter-based model for the notification task. The first step of the modeling activity consists of establishing the task-level *IRC* values for the design model (Norman, 1986), which characterize the notification objectives of the system. Designers inexperienced with the critical parameters associated with the domain of notification systems can take advantage of an *IRC* Calculation Wizard to obtain the desirable levels for their target system. By inquiring designers about the notification behavior of the target system along with typical users' expectations and benefits resulting from the notification, this modeling assistant enables designers to obtain accurate estimates for the desirable critical-parameter specifications (Chewar et al., 2004b; Chewar et al., 2004a). Designers presenting expertise with the domain can circumnavigate the module and specify the numerical value of this trait directly. After having established the targeted task-level critical-parameter values, designers specify the generalized tasks that most adequately model the primary tasks commonly associated with their envisioned system as well as indicate both visual and interactive design concerns associated to the target system's specific surroundings.

Next, the requirements tool presents designers with a standardized starting point for their task-modeling activity in the form of a generic task-model template. In particular, the tool presents all of the available templates corresponding to the class of systems associated with the desirable critical-parameter levels previously established. Allowing this filtering per meta-task genre increases the scalability of the tool and facilitates the identification of a proper template. Designers can select and base their task-modeling activity from a standardized template exhibiting the adequate meta-task. The tool then provides designers with all of the available task models instantiated from the selected template during the requirements phase of previous projects. However, if none of the generic templates adequately fit, designers can pursue the construction of a new one. The first step of this task model template creation process entails establishing the connectivity for the lower levels of the hierarchy through the specification of the governing critical-parameter components constituting the ideal meta-task genre corresponding to the set desirable task-level critical-parameter specifications for each of Norman's six stages of action. To facilitate subsequent task-model template reuse, designers can textually describe the behavioral objective of the notification upon users and their associated primary task modeled by the hierarchic archetype. The template creation process proceeds with the specification of the applicable subtasks and psychological factors for each stage of action. The applicable subtasks for the generic model are extracted from the stage-of-action breakdown of Sutcliffe's generic tasks (Montabert, 2006), while the psychological factors

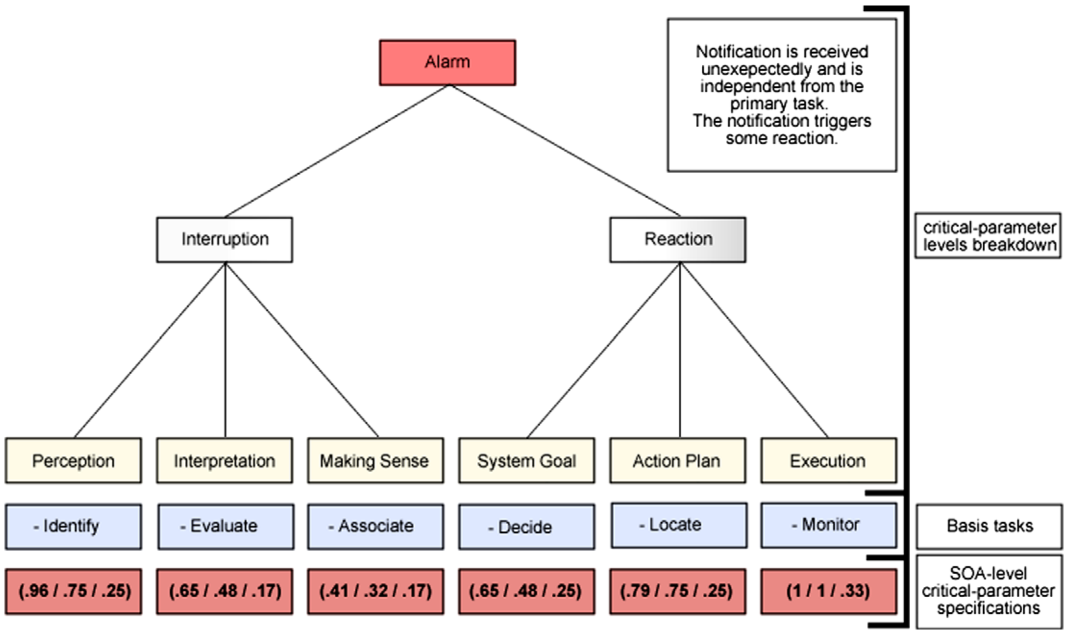


Fig. 3. Screenshot of the requirements tool summary page, which offers a comprehensive view of the work-product sets of the requirements process (i.e. problem and activity scenarios, desirable task-level critical-parameter values, critical-parameter decomposition pattern, stage-of-action scenarios, basic tasks, targeted SOA-level critical-parameter values, and claims IDs).

intervening in the *IRC* equations are estimated using a five-point scale (Chewar et al., 2004b). The tool then generates the SOA-level critical-parameter values and commits the newly constructed task-model template to the knowledge repository for consecutive reuse.

Following the selection of a generalized template as a basis for the modeling activity, the requirements tool provides an array of available instantiated task models. These task models result from the instantiation of the selected template through previous projects. To further pattern the task-model elaboration, reuse of such instantiated task models offers designers a more refined abstraction level, benefiting the effectiveness of the reuse solution (Krueger, 1992). Designers can select a declination of the generic task-model template that presents similar task-level critical-parameter objectives, and directly edit the task-model attributes to create a precise representation of their notification task. However, if none of the template digressions' targeted critical-parameter levels adequately suit, designers can pursue the construction of a new one. The first step of the instantiation practice involves, for each stage of action, a selection of the basic tasks available from the parent template that constitute the activity. Furthermore, because observations revealed that designers tend to refer to an activity instead of describing meticulously its constitution during the scenario-based domain analysis, leaving broad unspecified areas, to leverage validity, the system implements the elaboration for each stage of action of a scenario that narrates the subtask conduction, both ensuring proper generic task selection and reinforcing scenario coverage. After the specification of these attributes for each of the six stages, the requirements tool generates the desirable SOA-level critical-parameter values and displays a summative view of the task-modeling activity. As a requirements analysis milestone, this summary page allows designers to comprehensively review, verify, and validate the work-product of their modeling enterprise before moving forward.

A claims characterization completes the hierarchic task-model elicitation. First, in the form of a recommendation system, the requirements tool suggests potentially relevant claim-sets associated with the reused task model through projects. Second, to extend the model with additional claims, designers can access the search interface of a claims library and utilize the work-product established during the hierarchic modeling phase (i.e. SOA-level *IRC* values and generic tasks) to extract relevant knowledge from the repository, or create new claims (Fabian et al., 2004). Claims associated with the task model are committed to the repository and will be recommended through the subsequent reuse of the instance. Fig. 3 presents a screenshot of the system summary page of the requirements tool.

5. Validation through user evaluations

How accessible is the implementation of such process to novice designers? Does our integrative approach to requirements engineering really have the potential to bring the best of each individual technique and hold the promises of effective reuse at the requirements phase? To answer these questions, we present the design, procedure, and results for two studies. The first is a feasibility study to examine whether a novice designer could understand and conduct an analysis. The second is a benefits assessment to ascertain the degree to which requirements are impacted by the use of the tool.

5.1. Feasibility study

To provide insight about the possibility for novice designers to conduct successfully the activity of the process as well as the usability of the infrastructure, we invited seven students enrolled in an undergraduate HCI introductory class to participate in a usability evaluation of our requirements tool. Most of these novice designers had been previously exposed in the course of their curriculum to scenario-based design and the concept of claims analysis, but none of them were familiar with any of the notions involved in our critical-parameter-based task-modeling approach (i.e. hierarchical task analysis, *IRC* critical parameters, Norman's stages of action, and Sutcliffe's generalized and generic tasks). The inexperience of these candidates with respect to the majority of the facets of our process provided the opportunity to educate them to a similar degree in our novel design approach, then assess the difficulty associated with the task-modeling activity and self-sufficiency of the tool in communicating the concepts involved.

5.1.1. Method

Each participant received a set of generic instructions about the procedure and format of the evaluation but received no training or presentation about either the requirements tool or its supported process prior to the beginning of the assessment. In fact, to obtain a genuine evaluation about the self-sufficiency of the tool for the successful conduct of the entailed activities, we wanted the participants to remain naïve with respect to the supported activities and concepts involved in the process. Designers received a root concept as well as a problem and activity scenario and accessed the requirements tool. During their progression through the system, participants completed a trans-test questionnaire. After the completion of their requirements analysis, each participant completed a post-test questionnaire. No time constraints were associated with the evaluation.

5.1.2. Results

Encompassing twenty-four questions relying on a seven-point Likert scale, the trans-test survey focused on specific aspects of each page such as the difficulty of introduced concepts, clarity of the instruction, and ability for the tool to assist the understanding of such concepts through the provision of sufficient help. Results from the trans-test survey revealed the difficulty of some of the concepts involved with the task-modeling activity but also indicate the ability of the help section to mitigate such difficulties (Montabert et al., 2005).

Associated to a seven-point Likert scale, the post-test questionnaire entailed a usability portion as well as a section targeting the comprehensive user understanding of the tool and of its supported activities. First, to evaluate the usability of the tool, we created and administered a questionnaire. The first set of questions, loosely based on Nielsen's usability heuristics (Molich and Nielsen, 1990), suggested that the requirements tool was indeed usable (individual results for each the heuristics are available in Montabert, 2006). Second, the remaining portion of the questionnaire evaluated participants' understanding and ability to complete the process. User ratings reflect a participants' understanding of the objective and value of the tool ($M = 5.71$, $SD = 0.45$), as well as an understanding of the terms, diagrams, and concepts presented in this tool ($M = 5.29$, $SD = 0.70$). Users rated themselves able to conduct a requirements analysis for the given scenario with the tool ($M = 6.00$, $SD = 0.76$). Furthermore, with respect to the sample scenario, an inquiry of participants' artifacts subsequent to the evaluation revealed the selection of the correct task-model template and basic tasks along with the extraction of relevant claims from the repository, reinforcing the validity of these subjective ratings.

5.2. Benefits assessment survey

We invited seven undergraduate students to participate in a user study. First, the study aimed at evaluating whether a requirements-engineering process merging scenario-based requirements analysis and critical-parameter-based task analysis may be effective in leveraging scenario quality and capturing the critical-parameter specifications of a system. Second, the survey assessed whether a reliance on scenarios, task models, and critical parameters may benefit requirements quality, introduce effective reuse at the requirements phase, and increase the design quality of the resulting artifact. Third, the study evaluated whether an integrative requirements-engineering process exhibits potential educational payoffs for an acceptable difficulty level.

These participants, in their senior year, were enrolled in a research seminar on notification-system design. We preferred for this second investigation participants having an intermediate experience level with respect to the concepts and domain investigated, as such designers present enough proficiency to evaluate knowledgeably the process in terms of usefulness while remaining candid enough to judge adequately the difficulty level. Although, the participants had no formal expertise with either the conduction of hierarchical task analysis and task modeling or the use of Norman's stages of action and Sutcliffe's generic and generalized tasks, these students had knowledge of scenario-based design, claims analysis, and *IRC* critical parameters through the design of successful notification systems. In fact, each surveyed user was a member of one of three design groups who developed systems upon the SeeVT framework (Sampat et al., 2005), an infrastructure that takes advantage of both wireless internet access and ubiquitous computing to enable the development of location-based notification systems.

5.2.1. Method

The participants were divided into their three design teams with each team tested independently. Each surveyed user received a set of generic instructions about the procedure and format of the evaluation but received no training or presentation about either the requirements tool or its supported process. In fact, we wanted to ensure that the participants of each design team would be naïve with respect to the supported requirements process as well as the supporting tool's user interface at the start of the evaluation. The participants first individually completed a demographic information and pre-test questionnaire. Because this user evaluation aims at comparing the quality of the requirements process supported by our tool as well as the quality of its subsequent deliverables with respect to the quality of these obtained solely from a traditional scenario-based requirements analysis, in order to draw conclusions about the benefits of using such infrastructure, each design team was then instructed to re-conduct the requirements analysis for each of its respective semester-long projects. In fact, asking participants to re-conduct a requirements analysis yields two upsides. On one hand, teams had originally followed a development process solely based on scenario-based design which allows for a direct comparison of the quality of the requirements process offered by our tool. On the other hand, teams had already conducted their preliminary work such as field studies and stakeholders analysis during the original requirements phase of their respective projects, which enables an evaluation of the process in a reasonable amount of time while maintaining some degree of realism despite the controlled nature of the environment. To increase the degree of realism further, the groups were instructed to conduct their requirements analysis in a collaborative fashion. The requirements tool was then loaded on a large screen display, acknowledged as the best avenue for the exchange of ideas within collaborative work settings (Elrod et al., 1992). After the completion of their requirements analysis, participants individually completed a post-test questionnaire. No time constraints were associated with the evaluation.

5.2.2. Results

The pre-test survey consisted of five questions associated to a five-point Likert scale which focused on assessing the experience level of evaluators with respect to the concepts of scenario-based design, notification-system design, *IRC* parameters and requirements engineering. To obtain a reference level, the pre-test questionnaire asked participants to rate the quality and validity of the initial requirements elicited during the original iteration of the SeeVT projects (Bhatia et al., 2006; Nair et al., 2006; Sciacchitano et al., 2006). The results of the pre-test questionnaire indicate that the participants presented a broad range of expertise levels with respect to each concept. As a whole, participants reported an average expertise level for scenario-based design, notification-system design, and *IRC* parameters. Participants revealed to be somewhat inexperienced with respect to engineering requirements. Moreover, each design team encompasses members exhibiting different expertise levels with respect to each concept (the complete breakdown of the team constituents is available in Montabert, 2006).

The post-test survey was associated to a five-point Likert scale and targeted both the difficulty level as well as the benefit assessment associated with the overall requirements-analysis process and its supporting infrastructure. Asked to rate the difficulty level of the entire process along with its interface implementation, participants rated the process as easy to conduct ($M = 4.25$, SD = 0.42).

The survey then evaluated the effectiveness of the critical-parameter-based task-modeling approach for leveraging scenario quality and critical-parameter capture, as well as the benefits upon requirements quality, reuse, and subsequent design artifact resulting from the use of our tool. Participants deemed each evaluated criteria as successful with average user ratings in excess of four and no observable trends between participants' experience levels and ratings despite the diverse level of competencies. In particular, the task-modeling activity was rated effective upon the quality of scenario descriptions ($M = 4.00$, SD = 0.58) and capture of desirable critical-parameter levels ($M = 4.57$, SD = 0.53). Furthermore, our integrative approach to requirements engineering was judged as beneficial for achieving quality requirements ($M = 4.43$, SD = 0.53), reuse ($M = 4.57$, SD = 0.53), and design quality for the resulting artifact ($M = 4.42$, SD = 0.49). Fig. 4 shows the mean and standard deviation of the users' ratings for each of these five questions.

To assess the potential for educational payoff resulting from the use of the tool, the post-test questionnaire instructed participants to rate their awareness level of the importance of requirements for project success after having completed their requirements engineering with the tool. Users rated themselves as aware ($M = 4.29$,

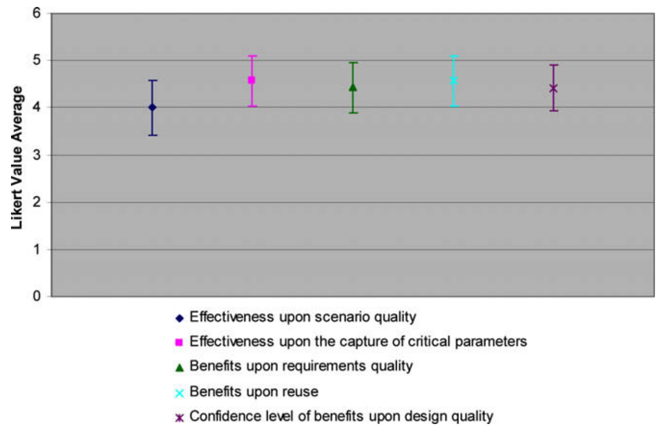


Fig. 4. User ratings of the effectiveness upon scenario quality and capture of critical parameters as well as benefits upon design, requirements, and reuse resulting from the use of the requirements tool (average ± standard deviation).

SD = 0.76). Furthermore, ratings suggest a pattern between users' experience levels and awareness levels, as users inexperienced with requirements-engineering practices exhibited higher awareness gains about the importance of the requirements phase.

A follow-up question to the pre-test survey instructed participants to evaluate, once again, the quality and validity of the requirements originally elicited for their SeeVT project now that they had re-conducted an entire requirements analysis for their project, but this time, using our process and supporting infrastructure. *T*-test shows that post-test quality and validity ratings of the requirements originally elicited ($M = 2.86$, SD = 0.38) are significantly smaller than the pre-test quality and validity ratings of the requirements originally elicited ($M = 3.57$, SD = 0.53) with $t(12) = 2.88$, $p = 0.014$.

Finally, comments about participants' experience with the requirements tool revealed that the system was enthusiastically received with eloquent statements reiterating the ratings captured throughout the questionnaire's benefits assessment portion. Verbatim et literatim comments such as "the tool is great for providing a base for successfully conducting requirements analysis" qualitatively confirmed the benefits upon requirements analysis, while "it is great to be able to conduct a requirement process based on the reuse of previous projects, it helps to get a feel for what we need to achieve" and "I did not know requirements were so critical for achieving good design. I really learned that by using this tool" respectively transcribed the incidence of reuse upon requirements and corroborated with the potential educational value resulting from the use of the tool.

5.3. Findings

Results from both user studies suggest the potential of our integrative approach to requirements analysis. First, results from our preliminary study indicate the ability for novice designers to successfully complete the activity entailed in our proposed requirements tool, judged usable. These results are key as they not only exhibit the usability of the tool, but also its suitability as a potential learning platform. Second, the results of our benefit assessment survey suggest that a requirements-analysis process combining scenario-based design and task analysis could be effective in leveraging scenario quality and capturing the critical-parameter requirements of a system.

In addition, it appears that the use of scenarios, task models, and critical parameters benefit requirements quality, promote reuse at the requirements phase, and increase the design quality of the resulting artifact. These patterns are reinforced by the significant decrease between users' self-assessments of the quality and validity of their prior projects' requirements recorded before and after the experiment, suggesting the requirements tool contributed to increase the value of the engineered requirements as well as designers' reference level and appreciation of requirements quality.

Moreover, as the criticality of the role of user involvement in developing quality requirements is replicate in the literature (Cheng and Atlee, 2007), the proposed requirements tool actively responds to this need by providing a more formal and structured approach that facilitates a better understanding of user needs through offering specific systemic components, implemented in a revised requirements analysis module, that support and foster more active and continuous user involvement through the requirements engineering process. As discussed, the revised interface of the requirements analysis module allows designers to directly bridge with the participatory negation module. It is our intent, through this systemic component, to leverage requirements quality, validity, and foster continuous user involvement beyond the early scenario states of the process. By facilitating this cross-module communication, designers will be more likely to engage user's participation in the requirements engineering process, whether for clarifying particular aspects of the behavior of the target system or ensuring stakeholder validation. The revised requirements analysis module enables designers to directly access the negotiation tool of the LINK-UP system. This provides a mechanism that supports the engagement of users in a process throughout which ideas can be exchanged to leverage the validity of the problem scenario. In addition if needed, designers can return to the scenario phase in an iterative fashion and revise their previous scenario descriptions; thus, enriching, overall, the requirements capture with a better understanding of user needs and requirements.

The evaluation also indicates the overall process' tasks and implementation are deemed effortless to conduct and acknowledges an increase in participants' awareness levels of the importance of requirements in project success, which implies a potential educational benefit resulting from this system with respect to novices with the domain of engineering requirements. The results of this study therefore provides support to our vision that the use of task models provides a good avenue for bridging the gap between scenarios and critical parameters while contributing toward reuse and forming a user-centric requirements-analysis process which put forth user involvement and quality requirements, conditions necessary for project success.

6. Conclusions and future work

This paper presents an integrative approach to requirements analysis which merges task modeling and critical parameters with scenario-based design to foster project success. Key characteristics of this approach to requirements analysis are a strong reliance on user involvement by following an overall scenario-based design practice, disambiguation and validation of requirements through structure and formalism via hierarchical task analysis and critical-parameter characterization, and extensive requirements knowledge transfer through the reliance on critical-parameter-based task models and claims. This approach shows potential to promote user involvement and leverage requirements accuracy at minimal expenditure—criteria recognized as critical for project success.

Through the grounding of our proposed user-centric and reuse-centric requirements-analysis approach within the domain of notification systems and consecutive implementation, two user studies recorded the feasibility and payoff associated with the use of such process and infrastructure. On one hand, our initial study confirmed the ability for novice designers to comprehend the activities involved with the critical-parameter-based task-modeling process, complete the analytic procedure successfully, and extract relevant reusable knowledge from a claims repository. On the other hand, the subsequent study validated the coupling of critical-parameter-based task modeling to scenario-based design improves scenario quality, facilitates the capture of critical-parameter specifications, and serves as a reuse catalyst while the emerging requirements-analysis process fosters requirements and design quality. The study also unveiled potential learning benefits associated with a potential deployment of such infrastructure as a teaching platform for requirements-engineering practices.

While evidence exists that suggests the efficacy of the tool, study findings should be balanced with the recognition of some challenges in the experimental design. Experimental parameters and controls were instituted that supported mitigating confounds; including allowing participants to leverage their existing front-end analysis work and allowing for the requirements-analysis process to be conducted in a collaborative fashion. Also, more than seven participants (comprising three design groups) were certainly desired, but the long-term participant commitment necessarily limited the number of participants. The experiences of the participants, collected over several months, led to insights that would have been difficult to procure from a larger group due to the needed time commitment. Finally and most notably, the academic setting and use of undergraduates as designers, regardless of their level of training, could not reflect fully an experienced design team. However, the composite expertise of the designers and their knowledge of the design domain of notification systems were essential in the use of our critical-parameter-based task-model approach and reflect the approach a company could take in adopting a critical-parameter-based approach to design—without the control and commitment costs that would be necessary in a controlled study in a corporate environment.

As to provide some understanding of the value and challenges associated with the use of our tool, questionnaires were designed and administered that featured loosely structured questions to more fully explore the participant's perceptions of our tool. Appropriate quantitative techniques were used to provide some insight into comparative differences in perceived quality of requirements generated with or without the use of our tool. Through a more formal analysis of participant responses (e.g. content analysis) was not conducted, themes were observed by the researchers that echoed the quantitative findings. These themes further highlight the potential value of the tool and will be explored in future research.

Finally, through the integration and adaptation of multiple requirements analysis techniques we provided the field with a comprehensive environment for eliciting requirements, while a capitalization on the task centricity of our approach via our critical-parameter-based task-modeling strategy not only offers the extraction of quality requirements, but also yields the generation of shareable domain-independent knowledge that can benefit the entire HCI community (Whittaker et al. 2000).

We recommend future research effort to deploy our integrative approach to other domains. In fact, we anticipate the extensions of such requirements process to other areas may yield the systematic identification of additional critical parameters, further validation and refinement of critical-parameter based task-modeling techniques, and increase both the quantity and reach of available claims that can be exchanged.

Because claims can characterize attributes within the problem space as well as attributes within the design space, claims offer the inclination for bridging these two worlds. Using our proposed approach in the design space, designers relate claims to their task models which become permanently associated to task models within the knowledge repository, making these problem claims readily available for reuse in other projects through task-model reuse. Once in the design space, designers will query the knowledge library in search of claims which adequately address the issues identified at the requirements phase. Continuing to extend the capability of the knowledge repository by preserving relationships between task models, problem claims and design claims may yield tremendous payoff for the design community.

Acknowledgements

The authors would like to give great thanks to the people who took part in our user evaluations. We would also like to thank Christa Chewar, Jacob Somervell, and Ali Ndiwalana for their comments on early versions of this work. Finally, we could like to thank the reviewers for their excellent and insightful comment on this paper.

References

Annett, J., 2003. Hierarchical task analysis. In: Holnagel, E. (Ed.), Handbook of Cognitive Task Design. Lawrence Erlbaum Associates, Mahwah, NJ, pp. 17–35 (Chapter 2).

Annett, J., Duncan, K.D., 1967. Task analysis and training design. Occupational Psychology 41, 211–221.

Bell, T.E., Thayer, T.A., 1997. Software requirements: are they really a problem? In: Proceedings of the 2nd International Conference on Software Engineering (ICSE-2), pp. 61–68.

Berry, D.M., Damian, D., Finkelstein, A., Gause, D., Hall, R., Simmons, E., Wassyng, A., 2005. To do or not to do: if the requirements engineering payoff is so good, why aren't companies doing it? In: Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE '05), p. 447.

Beyer, H., Holtzblatt, K., 1999. Contextual design. Interactions 6 (1), 32–42.

Bhatia, S., Dahn, C., Lee, J.C., Sampat, M., McCrickard, D.S., 2006. VTAssist – a location-based feedback notification system for the disabled. In: Proceedings of the 44th Annual ACM Southeast Regional Conference (ACMSE '06), pp. 512–517.

Boehm, B.W., 1981. Software Engineering Economics. Prentice Hall, Upper Saddle River, NJ.

Brooks, F.P., 1987. No silver bullet: essence and accidents of software engineering. IEEE Computer 20 (4), 10–19.

Carroll, J.M., Rosson, M.B., 1992. Getting around the task-artifact cycle: how to make claims and design by scenario. ACM Transaction on Information Systems 10, 181–212.

Cheng, B.H.C., Atlee, J.M., 2007. Research directions in requirements engineering. FOSE 2007, 285–303.

Chewar, C.M., Bachetti, E., McCrickard, D.S., Booker, J., 2004. Automating a design reuse facility with critical parameters: lessons learned in developing the LINK-UP system. In: Proceedings of the 2004 International Conference on Computer-Aided Design of User Interfaces (CADUI '04), pp. 236–247.

Chewar, C.M., McCrickard, D.S., Sutcliffe, A.G., 2004. Unpacking critical parameters for interface design: evaluating notification systems with the *IRC* framework. In: Proceedings of the 2004 Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques (DIS '04), pp. 279–288.

Diaper, D., 2002. Scenarios and task analysis. Interacting with Computers 14 (4), 379–395.

Elrod, S., Bruce, R., Gold, R., Goldberg, D., Halasz, F., Janssen, W., Lee, D., McCall, K., Pederson, E., Pier, K., Tang, J., Welch, B., 1992. Liveboard: a large interactive display supporting group meetings, presentations, and remote collaboration. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92), pp. 599–607.

Estublier, J., Vega, G., 2005. Reuse and variability in large software applications. In: Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 316–325.

European Software Institute, 1996. European software process improvement training Initiative (ESPITI) project: European user survey analysis. Technical Report ESI-1996-TR95104, European Software Institute.

Fabian, A., Felton, D., Grant, M., Montabert, C., Pious, K., Rashidi, N., Tarpley III, A. R., Taylor, N., Chewar, C.M., McCrickard, D.S., 2004. Designing the claims reuse library: validating classification methods for notification systems. In: Proceedings of the 42nd Annual ACM Southeast Regional Conference (ACMSE '04), pp. 357–362.

Gould, J.D., Lewis, C., 1985. Designing for usability: key principles and what designers think. Communications of the ACM 28, 300–311.

Grady, R.B., 1997. Successful Software Process Improvement. Prentice Hall, Englewood Cliffs, NJ.

Holtzblatt, K., Beyer, H.R., 1995. Requirements gathering: the human factor. Communications of the ACM 38 (5), 31–32.

Ishii, H., Ullmer, B., 1997. Tangible bits: towards seamless interfaces between people, bits, and atoms. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '97), pp. 234–241.

Krueger, C.W., 1992. Software reuse. ACM Computing Surveys 24 (2), 131–184.

Kujala, S., Kauppinen, M., Lehtola, L., Kojo, T., 2005. The role of user involvement in requirements quality and project success. In: Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE '05), pp. 75–84.

Lafrenière, D., 1996. CUTA: a simple, practical, low-cost approach to task analysis. Interactions 3 (5), 35–39.

Lim, W.C., 1994. Effects of reuse on quality, productivity, and economics. IEEE Software 11 (5), 23–30.

Lim, K.Y., Long, J., 1994. The MUSE Method for Usability Engineering. Cambridge University Press, Cambridge.

Martin, J., 1984. An Information Systems Manifesto. Prentice Hall, Paramus, NJ.

Matsumoto, Y., 1982. Software education in an industry. In: Proceedings of the 6th Annual International Computer Software and Applications Conference (COMPSA 1982), pp. 92–94.

Matsumoto, Y., 1993. Experiences from software reuse in industrial process control applications. In: Proceedings of Advances in Software Reuse: Selected Papers from the 2nd International Workshop on Software Reusability, pp. 186–195.

McCrickard, D.S., Chewar, C.M., 2003. Attentive user interfaces: attuning notification design to user goals and attention costs. Communications of the ACM 46 (3), 67–72.

McCrickard, D.S., Chewar, C.M., Somervell, J.P., Ndiwalana, A., 2003. A model for notification systems evaluation – assessing user goals for multitasking activity. ACM Transactions on Computer–Human Interaction (TOCHI) 10 (4), 312–338.

Mirel, B., 2000. Product, process, and profit: the politics of usability in a software venture. ACM Journal of Computer Documentation (JCD) 24 (4), 185–203.

Molich, R., Nielsen, J., 1990. Improving a human–computer dialogue: what designers know about traditional interface design. Communication of the ACM 33 (3), 338–348.

Montabert, C., 2006. Supporting requirements reuse in a user-centric design framework through task modeling and critical parameters. M.S. Thesis, Virginia Polytechnic Institute and State University.

Montabert, C., McCrickard, D.S., 2006. Task models, scenarios, and critical parameters: toward the establishment of an effective infrastructure for reuse-centric requirements analysis. In: Proceedings of the 3rd International Conference on Cybernetics and Information Technologies, Systems and Applications (CITSA 2006), pp. 186–191.

Montabert, C., Bussert, D., Gifford, S.S., Chewar, C.M., McCrickard, D.S., 2005. Supporting requirements reuse in notification systems design through task modeling. In: Proceedings of 11th International Conference on Human–Computer Interaction (HCII '05).

Nair, S., Kumar, A., Sampat, M., Lee, J.C., McCrickard, D.S., 2006. Alumni campus tour: capturing the fourth dimension in location based notification systems. In: Proceedings of the 44th Annual ACM Southeast Regional Conference (ACMSE '06), pp. 500–505.

Newman, W.M., 1997. Better or just different? On the benefits of designing interactive systems in terms of critical parameters. In: Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques (DIS '97), pp. 239–245.

Newman, W.M., Taylor, A.S., Dance, C.R., Taylor, S.A., 2000. Performance targets, models and innovation in interactive systems design. In: Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques (DIS '00), pp. 381–387.

Norman, D.A., 1986. Cognitive engineering. In: Norman, D.A., Draper, S.W. (Eds.), User Centered Systems Design: New Perspectives on Human–Computer Interaction. Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 31–61.

Paternò, F., Mancini, C., 1999. Late-breaking results: developing task models from informational scenarios. In: CHI '99 Extended Abstracts on Human factors in Computing Systems 1, pp. 228–229.

Paternò, F., Mancini, C., Meniconi, S., 1997. Engineering task models. In: Proceedings of the 3rd IEEE International Conference on Engineering of Complex Computer Systems, pp. 69–67.

Payne, C., Algood, C.F., Chewar, C.M., Holbrook, C., McCrickard, D.S., 2003. Generalizing interface design knowledge: lessons learned from developing a claims library. In: Proceedings of the 2003 IEEE International Conference on Information Reuse and Integration (IRI '03), pp. 362–369.

Richardson, J., Ormerod, T.C., Shepherd, A., 1998. The role of task analysis in capturing requirements for interface design. Interacting with Computers 9 (4), 367–384.

Rosson, M.B., Carroll, J.M., 2002. Usability Engineering: Scenario-Based Development of Human–Computer Interaction. Academic Press, San Diego, CA.

Sampat, M., Kumar, A., Prakash, A., McCrickard, D.S., 2005. Increasing understanding of a new environment using location-based notification systems. In: Poster paper in Proceedings of 11th International Conference on Human–Computer Interaction (HCII '05), auxiliary CD-ROM proceedings.

Sciacchitano, B., Cerwinski, C., Brown, I., Sampat, M., Lee, J.C., McCrickard, D.S., 2006. Intelligent library navigation using location-aware systems. In: Proceedings of the 44th Annual ACM Southeast Regional Conference (ACMSE '06), pp. 371–376.

Souchon, N., Limbourg, Q., Vanderdonckt, J., 2002. Task modeling in multiple contexts of use. In: Proceedings of Interactive Systems. Design, Specification, and Verification (DSV-IS 2002), pp. 59–73.

Sutcliffe, A., 1997. A technique combination approach to requirements engineering. In: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering, pp. 65–74.

Sutcliffe, A., 2000. On the effective use and reuse of HCI knowledge. ACM Transaction on Computer–Human Interaction (TOCHI) 7 (2), 197–221.

Sutcliffe, A., 2002. The Domain Theory: Patterns for Knowledge and Software Reuse. Lawrence Erlbaum Associates, Mahwah, NJ.

Sutcliffe, A., Carroll, J.M., 1999. Designing claims for reuse in interactive systems design. International Journal of Human–Computer Studies 50 (3), 213–241.

Taylor, F., 1991. Scientific Management. Harper & Row, New York.

Thayer, R.H., Dorfman, M., 1997. Software Requirements Engineering. Wiley-IEEE Computer Society Press, Los Alamitos, CA.

The Standish Group, 1994. The CHAOS report. <http://www.standishgroup.com/sample_research/chaos_1994_1.php> (retrieved 07.09.05).

van Dantzich, M., Robbins, D., Horvitz, E., Czerwinski, M., 2002. Scope: providing awareness of multiple notifications at a glance. In: Proceedings of the 6th International Working Conference on Advanced Visual Interfaces (AVI '02).

van Lamsweerde, A., 2000. Requirements engineering in the year 00: a research perspective. In: Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000), pp. 5–19.

Whittaker, S., Terveen, L., Nardi, B.A., 2000. Let's stop pushing the envelope and start addressing it: a reference task agenda for HCI. Human–Computer Interaction 15, 75–106.

Wiegers, K.E., 2003. Software Requirements. Microsoft Press, Redmond, WA.

Wilson, S., Bekker, M., Johnson, P. Johnson, H., 1997. Helping and hindering user involvement—A tale of everyday design. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '97), pp. 178–185.

Wurdel, M., Forbrig, P., Radhakrishnan, T., Sinnig, D., 2007. Patterns for task- and dialog-modeling. In: Proceedings of the 12th International Human Computer Interaction International Conference (HCII 2007), pp. 1226–1235.

Zand, M.K., Samadzadeh, M.H., 1994. Software reuse: issues and perspectives. IEEE Potentials 13 (3), 15–19.