Task Models, Scenarios, and Critical Parameters: Toward the Establishment of an Effective Infrastructure for Reuse-centric Requirements Analysis

Cyril MONTABERT
Center for HCI and Department of Computer Science, Virginia Polytechnic Institute and State University
Blacksburg, VA 24061, USA

and

D. Scott MCCRICKARD
Center for HCI and Department of Computer Science, Virginia Polytechnic Institute and State University
Blacksburg, VA 24061, USA

ABSTRACT

This paper presents the elaboration and implementation of a requirements-analysis process that unites task models, scenarios, and critical parameters to exploit and generate reusable knowledge at the requirements phase. Through the deployment of a critical-parameter-based approach to task modeling, the process yields the establishment of an integrative and formalized model issued from scenarios that can be used for requirements characterization. Furthermore, not only can this entity serve as interface to a knowledge repository relying on a critical-parameter-based taxonomy to support reuse but its characterization in terms of critical parameters also allows the model to constitute a broader reuse solution. We discuss our vision for a user-centric and reuse-centric approach to requirements analysis, present previous efforts implicated with this line of work, and state the revisions brought to extend the reuse potential and effectiveness of a previous iteration of a requirements tool implementing such process. Finally, the paper describes the sequence and nature of the activities involved with the conduct of our proposed requirements-analysis technique.

Keywords: Requirements, Task Modeling, Scenario-based Design, Critical Parameters, Reuse, Claims.

1. INTRODUCTION

Extensive surveys and literature *corpus* have identified the costly nature of lack of user involvement as well as the pervasiveness of errors in requirements specifications in the development of software projects [21, 6] On one hand, the economical constraints associated with most ventures and the viscosity associated with managerial decisions both restrain user involvement [24, 15] On the other hand, the attainment of requirements which capture users' real needs is a difficult assignment for designers to undertake, as requirements do not emerge naturally [2]. Because project failure results in tremendous costs, time, and effort, there is a crucial need for the establishment of a requirements-analysis infrastructure that promotes user involvement and ensures requirements veracity at low expenditures.

2. AN INTEGRATIVE APPROACH TO REQUIREMENTS ANALYSIS

To address the key attributes of user involvement, requirements quality, and low costs, we advocate the integration of alternative yet complementary techniques to engineering requirements. The process we are proposing is an integration of scenario-based design, task modeling, and critical parameters.

User-centric Approach to Quality Requirements

A reliance on user-centric approaches such as scenario-based design offers both the flexibility and accessibility needed for attaining stakeholders' involvement at minimal expenditure [20]. However, although scenario-based design has been acclaimed by many, it still presents limitations, such as recurrent bias and potential lack of coverage, which hinder its self-sufficiency as a preferred method for engineering requirements [5]. To mitigate these downsides, we recommend pairing the practice with a formal method such as hierarchical task analysis [1]. Through an explicit enumeration of the low-level tasks and activities a user can perform with a system, the resulting task models formalize and disambiguate users' tasks narrated in scenarios, addressing scenario bias, ensuring proper coverage, and leveraging an in-depth understanding of a usage situation. Finally, establishing formal metrics early in the design process, using a technique like critical parameters, enables designers to capture design objectives through a characterization of tasks, as the success or failure of software projects lay within the degree to which the targeted parameter values are reached [17]. An integrative requirements-analysis approach issued from scenario-based design, task modeling, and critical parameters shows potential to capitalize on the benefits of each individual methodology.

Reuse-centric Approach to Quality Requirements

Knowledge reuse is a key driver of quality at minimal resource expenditure. In fact, numerous accounts of the reuse potential upon increased productivity, reduced costs, and improved quality can be found in literature [13, 14, 10]. Although reuse payoffs can be maximized through an early introduction in the development cycle [22], minimal attention has been devoted to the building of supporting requirements reuse infrastructures and no real progress has been made [9]. A reliance on scenario-based design facilitates knowledge reuse through the generation of design claims. These reusable knowledge components explicitly formulate the upsides and downsides associated with the usage of a design feature. Previous endeavors have shown that critical parameters can provide an effective nomenclature for knowledge reuse indexing [7]. Unfortunately, scenario-based design does not offer sufficient built-in and explicit techniques needed for capturing the desirable critical-parameter levels, and consequently does not alone offer a systematic reuse mechanism. When conducting a hierarchical task analysis, designers systematically decompose the activity into basic tasks involved in a system interaction. Because the tasks and resulting subtasks objectives can be characterized in terms of critical parameters, the use of task models can bridge the gap between scenarios and critical parameters and reinforce the acquirement of taxonomic levels, decisive steps toward systematic and effective claims reuse [8]. However, task models should not be regarded solely as a reuse mediator. In fact, because tasks present some degree of independence with respect to application domains, they also constitute a shared body of knowledge that can be core to a reuse solution which can rely on a critical-parameter-based taxonomic scheme [25]. An integrative requirements-analysis approach issued from scenario-based design, task modeling, and critical parameters shows potential in addressing the dire need for an effective requirements reuse solution.

3. ESTABLISHMENT OF AN INFRASTRUSTURE

Prior to the establishment of a generic reuse infrastructure for requirements analysis, our first challenge was to identify a design space and knowledge storage approach. The increase of multitasking activity and pervasive thirst for digital information has led to the emergence of the domain of notification systems. Immerged into divided-attention environments, this class of systems aims at disseminating information effectively without introducing unwanted interruption to a primary task. The domain of notification systems exhibits valuable attributes for the purpose of our work because of the clearly identified and recognized critical parameters—Interruption, Reaction, and Comprehension (IRC)—which characterize the design space [12], the sensitivity upon the success of such systems to desirable critical-parameter specifications, and the auxiliary development infrastructures the framework already offers. Finally, because of the nature of this genre of software agent, it is possible to assimilate the objective of an entire system to the objective of their supported notification task.

A Computer-aided Design Tool Suite

The LINK-UP system is a tool suite aimed at providing structure and support to the design of notification systems [3]. Constituted of five primary modules—requirements tool, claims repository, negotiation tool, analytical tool, and empirical testing—the suite exhibits a development cycle compatible with scenario-based design. The work presented here focuses on the requirements tool and claims repository. Prior efforts have led to the development of the claims library [19, 7]. Within the database, the claim structure consists of a number of design attributes (i.e. title, description, upsides, downsides, design issues, associated scenario, artifact, IRC value) while the classification method relies on generalized tasks, generic tasks, and IRC value [23].

Critical-parameter-based Task Models

To aid with the formulation of critical-parameter levels in the design of interactive systems and bridge the abstraction gap between scenarios and critical-parameters, Montabert *et al.* introduced a critical-parameter-based approach to task modeling. Because a high-level task confines a user interaction, it can be decomposed in terms of stages of action [18]. As Sutcliffe's generic tasks describe simple unit procedures that are carried out to accomplish a single goal as well as offer the same granularity level than Norman's stages of action, these basic tasks can subsequently be used to characterize the cognitive and/or physical activity that occurs at each stage [23]. Finally, while critical parameters encompass the criteria for success of an interaction, these levels may not be persistent throughout each of the six stages of action constituting the activity but may rather translate into a particular sequence of critical-parameter objectives. This task modeling technique entails a systematic hierarchic decomposition of tasks into stages of action constituents along with a simultaneous decomposition of their critical-parameter characterizations [16].

Task-model Approach to Requirements Reuse

Because claims describe the key features of an activity, associating claims to each stage of action enables the hierarchic task model to embody the task requirements. Montabert *et al.* introduced the implementation of a procedure for requirements reuse *via* critical-parameter-based task modeling and claims reuse. Through the selection of a generic task-model template exhibiting adequate sequence of critical-parameter objectives with respect to the targeted task, designers customize each level of the hierarchic model to accurately represent the desirable task, not only making of this model a

formalization of requirements but also a reuse mediator. In fact, the structure associated with the constitution of such task model provides assistance to designers for the capture of the taxonomic attributes involved in claims search, fueling the extraction of relevant claims from the library. Results from a prior study unveiled some deficiencies but confirmed the ability for novice designers to successfully conduct a requirements-analysis process centered on task modeling and critical parameters, as well as obtain relevant information from the claims library [16].

4. REQUIREMENTS TOOL REFINEMENTS

During a six-month period, the requirements tool went through an iterative refinement process based on a rapid-prototyping approach paired with informal cognitive walkthroughs and heuristic evaluations, which lead to the identification of additional difficulties and entailed modifications to the requirements process.

Implementation of a Design-for-Reuse Mechanism
Although the first iteration of our infrastructure followed a design-by-reuse approach through the reuse of generalized task-model templates, recommended claims-sets, and claims, it is necessary for the requirements tool to implement an additional and intertwined design-for-reuse strategy for the reuse solution to be viable. The first facet in the implementation of a design-for-reuse strategy is the realization of a template creation process. In fact, for the tool to grow effectively and attain self-sufficiency, designers have the ability to extend the amount of available task-model templates by generating new ones. The second facet is the preservation of task models issued from generalized templates. In fact, to create a task model that depicts precisely the breakdown of the critical-parameter objectives per stage of action, applicable subtasks, and related claims for the project, designers customize preset templates. To capitalize on these efforts, these specified attributes and associated claims-sets become permanently associated with the instantiated task model to form a packet of design knowledge available for reuse in subsequent projects. Allowing designers to reuse preset templates and instantiated task models issued from previous projects not only leverages validity and increases the reuse potential of the tool as an additional level of ready-at-hand customizable reuse entity becomes available but also contributes to creating reusable components that populate the knowledge repository. The implementation of such mechanism makes the reuse-centered infrastructure autonomous; addressing the design-by-reuse design-for-reuse paradigm—key to reuse success.

Introduction of Stage-of-action Scenarios
We propose the introduction of the concept of *stage-of-action scenario* to describe the required nature and behavior of a task for each stage of Norman's model of action. In fact, we observed that when writing scenarios, the majority of designers mainly refer to the occurrence of tasks within scenario descriptions instead of describing the actual nature, behavior, and constituents of such task events as well as associated impact with respect to users' high-level objective. This negligence introduces major unspecified attributes which hinder any potential requirements elicitation and subsequent user validation, leaving these unspecified considerations up to the designers during the design phase. Overlooking such requirements may lead to the implementation of a system that does not properly captures users' expectations, a seed for project disaster. By encouraging designers to explicitly describe tasks' constituents in terms of subtasks within stage-of-action scenarios, we can leverage designers understanding of users' needs, enable user validation, and promote requirements quality. Finally, these stage-of-action scenarios are associated with the stages of action of the instantiated task-model template and become available for reuse in consecutive projects.

Integration of Task-model-dependant Stage-of-action-level Critical-parameter Levels
Because the recommended stage-of-action-level (SOA-level) critical-parameter values are associated to task-model templates which characterize the ideal system-level IRC values of a notification class, for these suggested values to be an accurate reflection of the system-level IRC breakdown per stage of action for an instantiated task model which may reflect a different high-level notification objective within such notification class, an algorithm had to be implemented. Such algorithm takes a fractional value of the recommended SOA-level IRC associated with the task-model template based on the percentage difference between the ideal system-level IRC values characterizing the generalized template and the design model's system-level IRC values. The implementation of such an algorithm within the requirements tool enables designers to obtain accurate critical-parameter-based search criteria for claims.

Iteration and Validation
To leverage requirements quality, validity, and foster continuous user involvement beyond the early scenario stages of the process, a revision of the requirements tool's interface enables designers to directly bridge with the negotiation tool of the LINK-UP system [3]. By facilitating cross module communication, designers will be more likely to engage users' participation in the process, whether for clarifying particular aspects of the behavior of the target system or ensuring stakeholders' validation. To further enhance the quality of our proposed requirements process and its resulting artifacts, the requirements tool enables users to return to previously completed stages and revise, in an iterative fashion, the work-product of each activity. Moreover, the system includes the addition of two summary pages. First, a task-

model summary page displays a complete view of the instantiated task model (i.e. task-level critical-parameter levels and task-model breakdown pattern, along with selected subtasks, stage-of-action scenarios, and SOA-level critical-parameter values for each stage of action). Second, the final step of the tool displays problem and activity scenarios with the addition of a summative overview of the complete instantiated task model and associated claims for the project. The implementation of these two summary pages provides users with a comprehensive view of the work-product of each conducted activity that constitutes the process, and as such, with an opportunity for verification and validation after which designers can either proceed or revisit and revise previous stages.

5. OVERVIEW OF THE REQUIREMENTS TOOL

After having conducted the preliminary inquiries necessary for the establishment of a root concept and studied the work practices in their natural settings, designers can access the requirement tool (Figure 1). An introductory page comprehensively describes the role of requirements analysis within the development cycle of interactive systems before introducing the module's key objectives and supported activities. The *raison d'être* for this page is to stress the importance of the requirements phase in software success.

The first step of the requirements tool is the formulation of problem scenarios. Crafted from user studies, problem scenarios enable designers to describe and analyze current work practices within the problem domain. The module instructs users about the format and objective of such problem scenarios. Designers specify a title for each problem scenario which characterizes actors, plot, and scope and enter a problem scenario description. Providing a title for each problem scenario enables project stakeholders to quickly index the nature of each scenario description and verify the scenario coverage. After having completed the problem scenario stage, designers move to the second step of the requirements tool, concerned with the establishment of activity scenarios. Describing users and their interaction with an envisioned system, activity scenarios enable designers to identify and extract the necessary activities that need to be supported by the target system. The module explains the format and objective of such activity scenarios after which designers can specify a title that encompasses actors, plot, and scope and enter an activity scenario description. Providing a title for each scenario enables project stakeholders to quickly index the nature of each description in order to verify the scenario coverage and adequacy.

Once the scenario phase has been completed, the tool allows for the establishment of the task-level critical-parameter values for the design model [18]. Novice designers can take advantage of an IRC Calculation Wizard to obtain the high-level notification goal of their target system. In the form of an auxiliary Flash sequence, the wizards asks designers numerous questions about the notification behavior of the intended system as well as typical users' expectations and benefits resulting from the notification. This tool enables designers to consistently obtain accurate estimates for the system-level IRC value of the design model [3, 4] Designers experienced in notification-system design can circumvent the IRC Calculation Wizard and specify the numerical value of this attribute directly. After having established the high-level notification goal the envisioned system needs to target, designers can indicate the generalized tasks that best characterize the primary tasks commonly associated with their target system as well as specify design concern related to the environment within which the target system is intended to progress. Design concerns include visual as well as interactive characteristics.

Based on the critical-parameter class corresponding to the task-level objective captured in the form of critical-parameter levels at the previous step, the requirements tool presents designers with all the available task-model templates that depict such critical-parameter meta-task. Designers then either select the task-model template that represents the targeted critical-parameter sequence of the notification interaction for the target system or create a new one if none adequately depicts such behavior. if the creation of a new template is pursued, designers are required to first indicate the connectivity for the lower level of the task-model template by specifying the governing critical-parameter component for each stage of action based on the ideal IRC value of the notification-system class corresponding to the system-level IRC value previously established. Designers can subsequently enter a textual description of the modeled critical-parameter behavioral objective encapsulated in the template. Once the basic connectivity for the task-model template has been established, for each of Norman's six stages of action, designers specify the applicable subtasks from the list of Sutcliffe's generic tasks [23] along with the psychological factors constituting the critical parameters using a five-point scale [4]. Based on designers' estimates for the psychological factors associated with each stage of action, the tool calculates the recommended SOA-level critical-parameter values for the template. The newly created task-model template is committed to the knowledge repository and becomes available for reuse.

Once a generalized task-model template has been selected, the tool displays all of the task models instantiated from the chosen template during previous projects. Designers either select to reuse a previously instantiated task model if one of the multiple declinations of the single template reflects similar high-level IRC value or create a new instance directly from the template

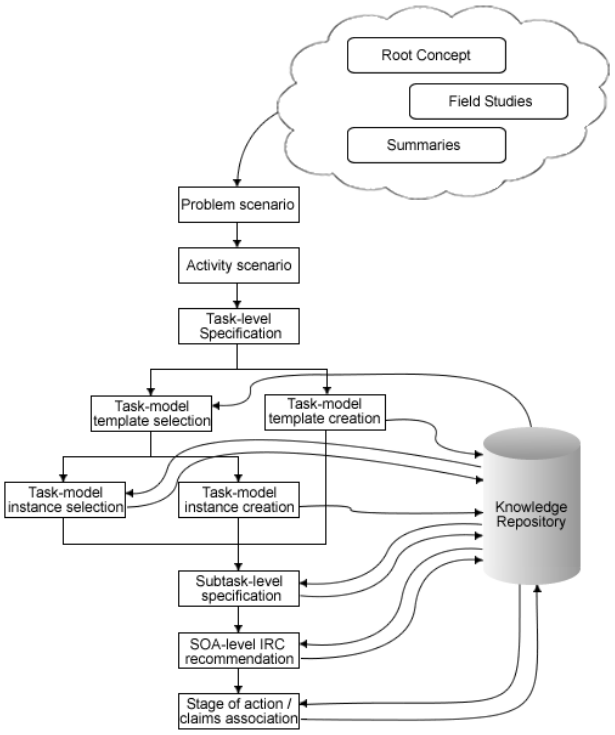

Figure 1: Flowchart of the requirements tool's supported activities[1] with an integrative view of Carroll's scenario-based requirements-analysis process as well as their relationship with the knowledge repository.

if none adequately express such targeted notification objectives. Once the task-model has been selected, the tool enables designers to indicate the subtask-model level of their task model. For each stage of action, designers select the applicable generic tasks available from the associated task-model template and provide a stage-of-action scenario accurately narrating the realization of the subtask. If designers base their work on a reused task-model instance, the tool displays the previously selected subtasks and provided stage-of-action scenarios, which enables designers to directly edit the content for each stage of action. After the extension of the task-model with this subtask-model level, the tool generates the desirable SOA-level IRC value which characterizes the notification objective of each subtask for each stage of action and displays a summary view for the project task model.

Finally, the module presents designers with recommended claims associated with a reused task model. Designers can access the Claims Reuse Library search interface, query the claims database using the generic tasks selected during the constitution of the subtask model as well as SOA-level IRC values recommended by the tool to obtain additional relevant claims, and associate the retrieved claims to the corresponding stages of action, creating an embodiment of the notification requirements for the hypothetical system. These relevant claims become associated to the stages of action of this particular task model and are stored in the knowledge repository to provide the foundation for the claims recommendation structure.

[1] For clarity, the iterative nature of activities was omitted.

6. CONCLUSIONS AND FUTURE WORK

By merging scenario-based design and task analysis, we anticipate that the implementation of our requirements analysis process will be effective in leveraging scenario quality and capturing the desirable critical-parameter levels of a system while the use of task models, scenarios, and critical parameters will benefit requirements quality, promote reuse at the requirements phase, and increase the design quality of the resulting design artifact at low expenditure.

Future work will be dedicated to assessing and validating the benefits upon requirements work-products resulting from a reliance on our proposed infrastructure. Furthermore, we also plan to integrate this method within a broader requirements analysis framework in order to assist designers in the formalization of every other facet of the requirements.

7. ACKNOWLEDGEMENTS

Thank you to the Virginia Tech ASPIRES program for partially funding this work and to the Virginia Tech Undergraduate Research in Computer Science (VTURCS) for their precious help and valuable feedback.

8. REFERENCES

[1] J. Annett, "Hierarchical Task Analysis", In E. Holnagel (Ed.), **Handbook of Cognitive Task Design**, Chapter 2 (pp. 17–35), Mahwah, NJ: Lawrence Erlbaum Associates, 2003.

[2] T.E. Bell and T.A. Thayer, "Software Requirements: Are They Really a Problem?", **Proc. 2nd International Conference on Software Engineering (ICSE-2)**, 1976, pp. 61–68.

[3] C.M. Chewar, E. Bachetti, D.S. McCrickard, and J. Booker, "Automating a Design Reuse Facility with Critical Parameters: Lessons Learned in Developing the LINK-UP System", **Proc. 2004 International Conference on Computer-Aided Design of User Interfaces (CADUI '04)**, 2004, pp. 236–247.

[4] C.M. Chewar, D.S. McCrickard, and A.G. Sutcliffe, "Unpacking Critical Parameters for Interface Design: Evaluating Notification Systems with the IRC

Framework", **Proc. 2004 Conference on Designing Interactive Systems (DIS '04)**, 2004, pp. 279–288.

[5] D. Diaper, "Scenarios and Task Analysis", **Interacting with Computers**, Vol. 14, No. 4, 2002, pp. 379–395.

[6] European Software Institute, "European Software Process Improvement Training Initiative (ESPITI) Project: European User Survey Analysis", **Technical Report ESI-1996-TR95104, European Software Institute**, 1996.

[7] A. Fabian, D. Felton, M. Grant, C. Montabert, K. Pious, N. Rashidi, A.R. Tarpley III, N. Taylor, C.M. Chewar, and D.S. McCrickard, "Designing the Claims Reuse Library: Validating Classification Methods for Notification Systems", **Proc. ACM Southeast Conference (ACMSE '04)**, 2004, pp. 357–362.

[8] C.W. Krueger, "Software Reuse". **ACM Computing Surveys**, Vol. 24, No. 2, 1992, pp. 131–184.

[9] A. van Lamsweerde, "Requirements Engineering in the Year 00: A Research Perspective", **Proc. 22nd International Conference on Software Engineering (ICSE 2000)**, 2000, pp. 5–19.

[10] W.C. Lim, "Effects of Reuse on Quality, Productivity, and Economics", **IEEE Software**, Vol. 11, No. 5, 1994, pp. 23–30.

[11] D.S. McCrickard and C.M. Chewar, "Attentive User Interfaces: Attuning Notification Design to User Goals and Attention Costs", **Communications of the ACM**, Vol. 46, No. 3, 2003, pp. 67–72.

[12] D.S. McCrickard, C.M. Chewar, J.P. Somervell, and A. Ndiwalana. "A Model for Notification Systems Evaluation—Assessing User Goals for Multitasking Activity". **ACM Transactions on Computer-Human Interaction (TOCHI)**, Vol. 10, No. 4, 2003, pp. 312–338.

[13] Y. Matsumoto, "Software Education in an Industry", **Proc. 6th Annual International Computer Software and Applications Conference (COMPSA 1982)**, 1982, pp. 92–94.

[14] Y. Matsumoto, "Experiences from Software Reuse in Industrial Process Control Applications", **Proc. Advances in Software Reuse: Selected Papers from the 2nd International Workshop on Software Reusability**, 1993, pp. 186–195.

[15] B. Mirel, "Product, Process, and Profit: the Politics of Usability in a Software Venture", **ACM Journal of Computer Documentation (JCD)**, Vol. 24, No. 4, 2000, pp. 185–203.

[16] C. Montabert, D. Bussert, S.S. Gifford, C.M. Chewar, and D.S. McCrickard, "Supporting Requirements Reuse in Notification Systems Design through Task Modeling", **Proc. 11th International Conference on Human-Computer Interaction (HCII '05)**, 2005.

[17] W.M. Newman, "Better or just different? On the Benefits of Designing Interactive Systems in Terms of Critical Parameters", **Proc. Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques (DIS '97)**, 1997, pp. 239–245.

[18] D.A. Norman, "Cognitive Engineering", In D.A. Norman and S.W. Draper (Eds.), **User Centered Systems Design: New Perspectives on Human-Computer Interaction** (pp. 31–61). Hillsdale, NJ: Lawrence Erlbaum Associates, 1986.

[19] C. Payne, C.F. Algood, C.M. Chewar, C. Holbrook, and D.S. McCrickard, "Generalizing Interface Design Knowledge: Lessons Learned from Developing a Claims Library", **Proc. 2003 IEEE International Conference on Information Reuse and Integration (IRI '03)**, 2003, pp. 362–369.

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

[21] The Standish Group. "The CHAOS Report", http://www.standishgroup.com/sample_research/chaos_1994_1.php, 1994.

[22] A. Sutcliffe, "On the Effective Use and Reuse of HCI knowledge", **ACM Transaction on Computer-Human Interaction (TOCHI)**, Vol. 7, No. 2, 2000, pp. 197–221.

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

[24] S. Wilson, M. Bekker, P. Johnson, and H. Johnson, "Helping and Hindering User Involvement—A Tale of Everyday Design", **Proc. SIGCHI Conference on Human Factors in Computing Systems (CHI '97)**, 1997, pp. 178–185.

[25] S. Whittaker, L. Terveen, and B.A. Nardi, "Let's Stop Pushing the Envelope and Start Addressing It: A Reference Task Agenda for HCI", **Human-Computer Interaction**, Vol. 15, 2000, pp. 75–106.