# Supporting Requirements Reuse in Notification Systems Design through Task Modeling

*Cyril Montabert, Dillon Bussert, Solomon S. Gifford, C. M. Chewar, and D. Scott McCrickard*

Center for Human Computer Interaction and Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061 USA
+1 540 231-7409
{cmontabe, dbussert, sgifford, cchewar, mccricks}@vt.edu

## Abstract

In hierarchical task analysis, designers study and decompose the tasks users perform while interacting with a system. Overlooking task analysis can have serious consequences on the design outcome. By making tasks clear to the designers, we can leverage an in-depth understanding of a usage scenario. Resulting from task analysis is a task model that explicitly enumerates the low-level tasks and activities a user can perform with a piece of software. Our first challenge was to identify a design space (notification systems) and knowledge storage approach (claims). By attaching claims to tasks in a hierarchical task analysis, the model creates a visualization of relationships, embodying the system requirements. Successfully adapting task modeling to model notification tasks shows promise in benefiting the requirements analysis of notification system design and promoting reuse. This paper describes a procedure and instantiation of hierarchical task analysis, to help designers identify the most important aspects to focus design and to support reuse of design knowledge within and between projects. The results of our usability study make us confident that users are able to understand the processes of task modeling if provided with sufficient help.

## 1 Introduction

The advent of new multitasking informational applications such as stock tickers, task bar icons, and in-vehicle displays is a response to users' needs to track information while not being unnecessary distracted from their primary task. These secondary software applications that fulfill user's information needs are referred to as *notification systems*, interfaces designed to disseminate valued information, in an efficient and effective manner, without introducing unwanted interruption to a primary task (McCrickard, & Chewar, 2003).

### 1.1 Notification Systems

The challenge that arises when designing notification systems is to accurately identify the key requirements that the system should satisfy. Designers need to determine whether and how much the user should be *interrupted*, whether the user has to stop what s/he is doing immediately or at a later time to *react* to the notification, and whether the notification system should support long-term *comprehension* of information. These types of systems use specially designed interfaces to specifically meet the *interruption*, *reaction*, and *comprehension* requirements of the system (McCrickard, & Chewar, 2003). Consequently, because the interfaces are *specially designed*, significant time is required for their development.

Most software engineers appreciate the idea of code reuse, as a time and cost saving mean. Similarly, when software designers talk about the client-server relationship or the pipe and filter paradigm, they are reusing design knowledge, such that the abstract model implies a list of predetermined design considerations. Because well-designed notification systems are designed around *interruption*, *reaction*, and *comprehension* (IRC) parameters, the challenge becomes how one can scientifically identify and pattern notification system knowledge in a reusable manner (McCrickard, & Chewar, 2003).

Time and other limited resources eliminate the option to design from scratch. Most importantly, doing so would involve revalidating and testing each design component, eschewing any potential benefit of what has been learned in the past (Carroll, 2000). The idea of reuse cannot be discarded due to the challenge.

## 1.2 Claims Library

One approach discussed in (Chewar, McCrickard & Sutcliffe, 2004), (Payne, Algood, Chewar, Holbrook & McCrickard, 2003) and (Chewar, Bachetti, McCrickard & Booker, 2004) to facilitate reuse is the implementation of a claims library. Sutcliffe (Sutcliffe, 2000) supported the idea of claims as a way to capture reusable design knowledge in *scenario-based design* (Rosson & Carroll, 2002). A claim is a statement describing a design characteristic with its associated benefits (upsides) and its weaknesses (downsides) (Chewar, Bachetti, McCrickard & Booker, 2004) (Figure 1).

Use of color to indicate a significant change in status of a condition the user is monitoring:
+ Use of the color change is an easy way to see if there is a status change or not.
- Repeated color changes (or flashing behavior) could become distracting if flashing is continuous.

**Figure 1: A claim example**

In order to index or relate claims to the requirements of a notification system, each claim is assigned an IRC value. For example, the above claim would be characterized as having a low interruption (as the user does not have to deviate from the primary task immediately), a low reaction (as the user does not have to respond to the new information immediately), and a low comprehension (as little information is conveyed by the color).

# 2 Task Modeling

In hierarchical task analysis, designers study and decompose the tasks users perform while interacting with a system. Resulting from task analysis is a task model that explicitly enumerates all the tasks and activities that a user can perform with a piece of software (Souchon, Limbourg & Vanderdonckt, 2002). Often displayed as a tree graph, a task model represents hierarchical and temporal relationships between tasks, leading to a visualization of how the software can be used (Mori, Paternò & Santoro, 2002). This type of model can be used to aid design by validating requirements and implementation as well as facilitating reuse of previous task models. Not only does this design technique provide guidance during the requirements analysis, but it also helps identify the most effective way a system should support these tasks.

The process of task analysis is often implicitly conducted. When reading usage scenario, people create an envisioned model based on assumptions, which often lead to communication problems among a team working with the same set of scenarios (Diaper, 2002). Consequently, overlooking task analysis can have serious consequences on the design outcome. In fact, not only might the system not support the right tasks, but it also is unlikely the system will support these correct sets of tasks in an effective way (Lewis & Rieman, 1994). The design model captures the developer's envisioned system (Norman, 1986). Making task analysis explicit with the integration of task models within scenario-based design to aid development of the design model can greatly benefit the overall design process of notification systems. In fact, by making tasks clear to the designers, we can leverage an in-depth understanding of a usage scenario and correct *scenario bias* (Carroll, 2000).

In addition, understanding the relationship among task models is the first step toward promoting design reuse. These task models detail the important tasks of a system, but the tasks are often similar and can be shared across many projects. A developer must discover the similarities among designs to ensure good design elements can be extracted and applied to new situations. For example, a task model for an alarm in one design is likely to be similar to a task model for an alarm in another design.

Because notification systems are closely related to human activity and the involvement in a primary task, the use of task models needs to be adapted to be fully effective with notification systems design. In fact, in our problem domain, slightly changing the way information is displayed can drastically influence the notification goal (McCrickard & Chewar, 2003), even though the task still consists of conveying information. Successfully adapting task modeling to model notification tasks can benefit the requirements analysis of notification system design, while promoting reuse. In addition, the use of task models with explicit subtasks provides designers with accurate search criteria for conducting claims analysis. In fact, it enables designers to query a claims library using a lower level of abstraction for the search criteria, which supports and facilitates claims reuse effectiveness.

# 3 Overview of the LINK-UP System

Currently under development, the LINK-UP system (Leveraging Integrated Notification Knowledge through Usability Parameters) is a web-based computer-aided design tool suite intended to better support the design of notification systems through reuse (Figure 2). More specifically, the system consists of five primary modules, including a requirements analysis tool and a claims library that support requirements analysis from a scenario-based approach. The current research work focuses on the Requirements Tool and the Claims Repository.

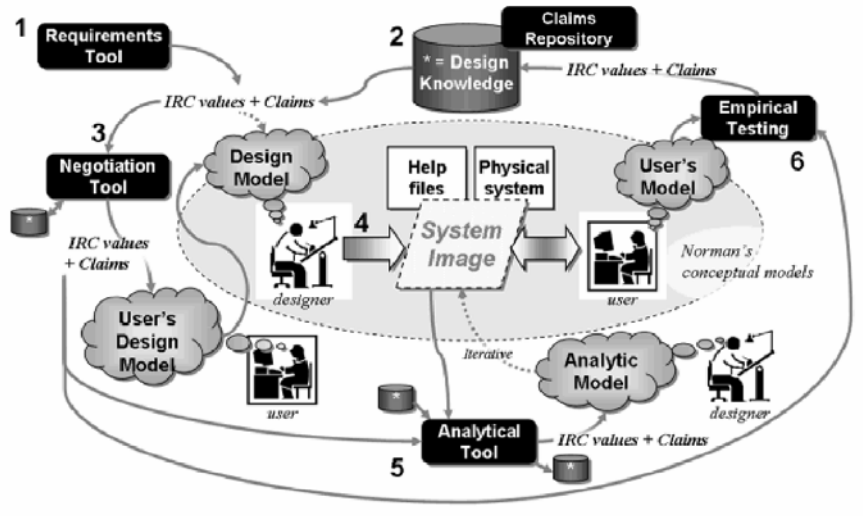


**Figure 2: A Vision of the LINK-UP System**

## 3.1 The Requirements Tool

The first tool used in the design process incorporated in the LINK-UP system is the Requirements Tool. It assists designers in analyzing and gathering user requirements to help with the elaboration of a *design model* (Norman, 1986). That is, the tool helps the designer discover and understand the underlying tasks, information, and activities to be supported, as well as the surrounding environment or *domain* (Sutcliffe, 2002). Designers are guided through specific steps to achieve their design model. This includes helping designers estimate the targeted *IRC value* of their notification system (McCrickard & Chewar, 2003), (Chewar, Bachetti, McCrickard & Booker, 2004).

## 3.2 The Claims Reuse Library

The claims reuse library is a web-based claims library consisting of design tokens or *claims* (Sutcliffe & Carroll, 1999), (Payne, Algood, Chewar, Holbrook & McCrickard, 2003), (Fabian et al., 2004). Each claim consists of a number of design attributes including title, description, upsides, downsides, design issues, associated scenario, artifact, and IRC value that describe the effect that some features will have on a user in an associated scenario. The claims classification within the repository relies on several parameters. The primary parameter is the IRC value, as it numerically captures the effect the claim will have on the notification system once implemented. Other parameters include the primary tasks (based on Sutcliffe's *generalized tasks* (Sutcliffe, 2002)), which provide insight about the tasks commonly executed by the user at the time the notification occurs; the *generic tasks* (Sutcliffe, 2002), which characterize what notification tasks are associated with the notification system; and finally the design concerns, which specify other constraints associated with the design.

Initial results based on classification via these parameters suggest that consistent values can be achieved with multiple users when adding claims to the repository (Fabian et al., 2004). These initial results are essential because to be fully effective in the design of notification systems, the system must either integrate or encourage design and knowledge reuse, by making it easy and reliable. These parameters support reuse successfully by providing users with accurate search criteria by which they can find relevant claims.

# 4 Requirements Analysis and Hierarchical Task Analysis in LINK-UP

The focus of this research work is the design and implementation of the Requirements Tool. The failure of many notification systems is the direct consequence of inaccurate estimates and erroneous assumptions made throughout the requirements analysis phase. Encouraging reuse during the requirements analysis phase would not only shorten the development time, but it would also guide the designer and reduce errors as new notification systems would be based on previous successful systems. The key challenge in assisting the design of notification systems is therefore to determine what approaches designers should follow to facilitate their design decisions while ensuring that their design would successfully meet its notification goals.

The solution presented is the use of claims to integrate requirements into task models. The claims describe how the system should look, feel, and act. By attaching claims to tasks in a hierarchical task analysis, the task model creates a visualization of relationships, embodying the system requirements. Additionally, the subtasks in the lowest level of the hierarchical task decomposition are classified by Norman's stages of action (Norman, 1986). The stages of action organize subtasks represented when users form cognitive relationships as they interact with a notification system. The stages of perception, interaction, and making sense allow users to understand the information the system communicates. The resulting stages of creating a system goal, an action plan, and its execution, allow users to interact, whether cognitively or physically, with the notification system to achieve their desired outcome.

In order to relate claims to subtasks, the IRC value of the entire system is calculated by asking the user a series of questions (Chewar, McCrickard & Sutcliffe, 2004). While the notification goal of an entire system is captured in its IRC value, it does not imply that the IRC value of every associated claim must match the IRC value of that system. Just as the hierarchical task analysis breaks down the tasks of the system into notification tasks (subtasks), the IRC value of an entire system can be broken down into more applicable sub-IRC values using the original IRC equations (Chewar, McCrickard & Sutcliffe, 2004). Each of these sub-IRC values characterizes the notification goals of the subtasks and can then be used to relate claims.

Using these notification tasks and sub-IRC values as search criteria for claims rather than the IRC value of the entire system provides a lower granularity, enabling the extraction of more accurate and relevant knowledge from the claims library, encouraging and facilitating reuse of the design knowledge encapsulated in the claims. Moreover, to further leverage reuse, creating templates of generic task models, along with related subtasks, provide designers with a standard starting point for requirements analysis that can be employed in new designs. By restricting the subtasks to Sutcliffe's generic tasks (Sutcliffe, 2002) and organizing them according to the stages of actions they fall into based on their pre and post-conditions, the subtasks in the templates represent a Lingua Franca (Erickson, 2000) by which designers can relate claims. Cast from previous designs, these templates enumerate the significant tasks of common applications. In fact, for a given notification system genre, depending upon how the system is meant to be used, the breakdown of the IRC value will affect the crossing of the stages of actions differently (Figure 3).

Further, claims associated with a subtask in one notification system are more likely to be relevant in another notification system sharing the same template. This relationship serves to provide designers with "suggested claims" based on the task model they selected.

## 4.1 Implementation

To put the above theory into practice, we integrated these ideas into our development of the Requirements Tool for LINK-UP. After having created a new project and added stakeholders, designers are prompted to create problem scenarios for their project. Following, a wizard allows them to obtain the IRC value for their notification system (Chewar, McCrickard & Sutcliffe, 2004). Based on the generated IRC value, the Requirements Tool presents the available list of task-model templates representing the notification system genre to be designed. We developed two task-model templates each for three notification system types—alarm, secondary display, and indicator—as they model three corners of the IRC framework (Chewar, McCrickard & Sutcliffe, 2004). Briefly, an alarm represents the pairing of high interruption with high reaction goals, an indicator focuses on high reaction, and high comprehension goals, while a secondary display characterizes only a high comprehension goal. Future versions will allow advanced designers to create or import new task models. Users can then select the notification tasks relevant to each stage of action (Figure 4). Once selected, the Requirements Tool provides designers with an interface, integrated with the claims library, by which they can associate claims with the task model according to the subtasks and indexed by sub-IRC values.

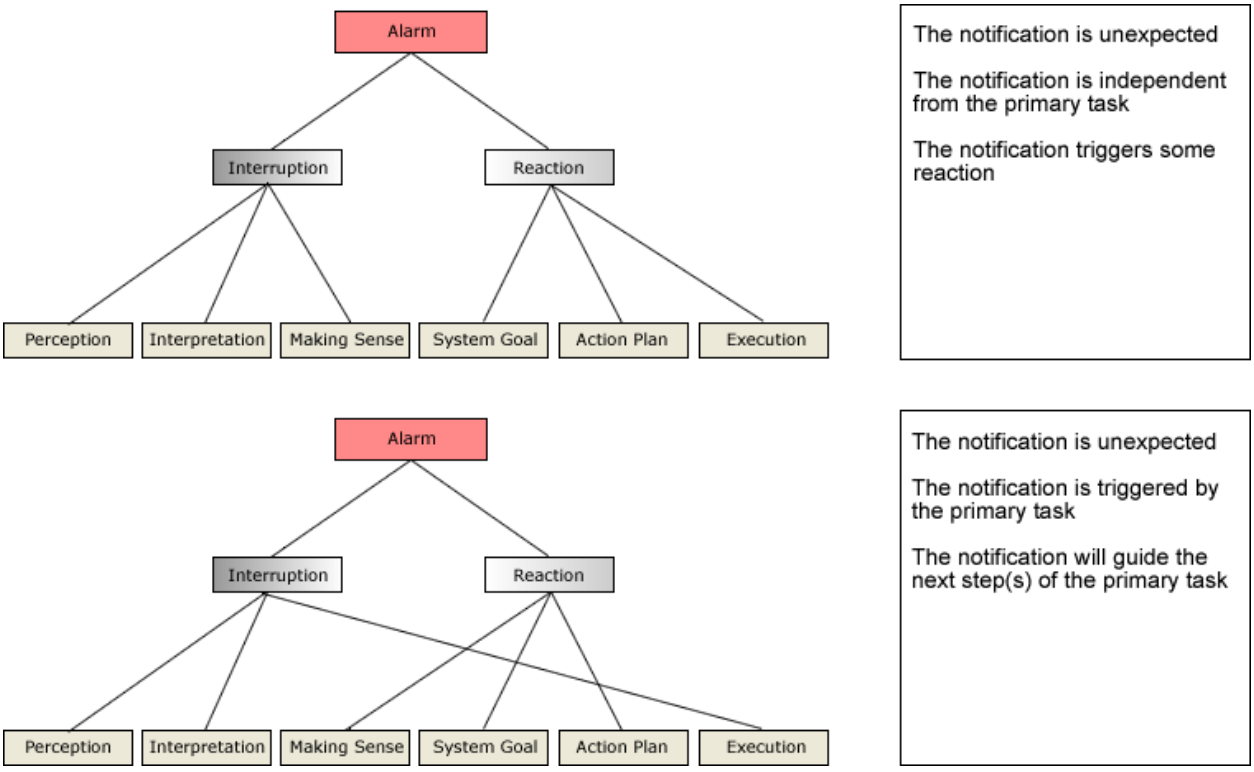


**Figure 3: Two task models for an alarm**

# 5 Usability Study

The goal of the interface is to support and encourage claims reuse in the requirements analysis phase. Experts analyzed and verified the correctness of the templates and classifications, but the system still needed to be tested in a real-world setting to determine if it had the potential to support requirements analysis for notification systems. Target users were not familiar with conducting hierarchical task analysis, estimating IRC values, and in some cases, even with claims. Consequently, the system had to provide enough guidance to educate users about the process, in order for them to successfully conduct a requirements analysis.

Our usability study focused on user understanding to verify that the help provided was actually beneficial. We opted for a heuristic evaluation to analyze the interface as this usability testing technique is easy to perform, inexpensive, and can be done with a few users. In addition, it provides clear and immediate feedback about the system. Each iteration uncovers potential areas for improvement which can be addressed before further iterations. A couple of rounds of informal testing with usability experts allowed some interfaces adjustments so that targeted users could navigate through the requirements process.

## 5.1 Procedure

To evaluate the interface with the target user group, we chose to test seven Computer Science students enrolled in an Introduction to Human-Computer Interaction class. In fact, previous studies have shown that the optimum cost-benefit ratio for heuristic testing is between three to seven evaluators (Nielsen, 1994). These students had some knowledge of scenario-based design and claims analysis, but had no previous experience with conducting a hierarchical task analysis or using Norman's stages of action. The students participated in a realistic walk-through of the interface using a sample scenario, as suggested by Nielsen: "If the system is domain-dependent and the evaluators are fairly naive with respect to the domain of the system, it will be necessary to assist the evaluators to enable them to use the interface. One approach that has been applied successfully is to supply the evaluators with a typical usage scenario, listing the various steps a user would take to perform a sample set of realistic tasks" (Nielsen, 1994).

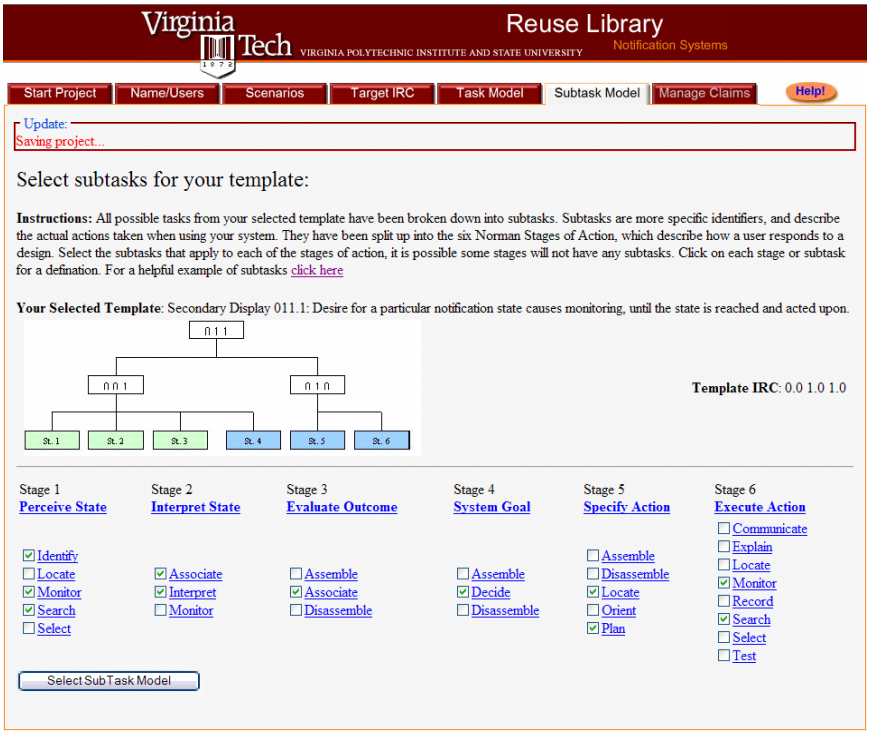


**Figure 4: Task model of a "Secondary Display, where a desire for a particular notification state causes monitoring, until the state is reached and acted upon." Subtasks are organized by Norman's stages of action, and filtered to be relevant to the presented template.**

To educate the users, the requirement analysis module provides significant help and instructions to explain each page in the process. We expected that users would not be able to understand all of the concepts used, but would be able to learn how to do requirements analysis as they went along by using the help provided.

Testers were presented a sample scenario and were asked to complete a requirements analysis using the system. To emulate realistic goals and tasks to be completed, the sample scenario instructed the participant to imagine a software company had given them the task of creating a notification system. During the testing, they explored every page in the interface while learning about the requirements analysis process.

As they were completing the necessary steps, they filled out a trans-test questionnaire to validate their understanding of each step of the module, to estimate the quality of the help provided, as well as the guidance offered by the directions. After completion of their analysis, the students filled out a post-test questionnaire. Students could give comprehensive feedback about their appreciation of the entire module, helping identify problems and suggesting key reengineering changes.

## 5.2 Results

After running the experiment, we analyzed the results of the trans-test and post-test questionnaires. Our Likert scale consisted of seven answers: strongly disagree, disagree, slightly disagree, neutral, slightly agree, agree, and strongly agree. Each of the answers was assigned a corresponding numeric value from zero to six, a response of "strongly disagree" having a value of zero, and a response of "strongly agree" having a value of six. We averaged the numeric values for each question and calculated the standard deviation.

### *5.2.1 Trans-Test Survey*

Consisting of twenty-four questions, the trans-test survey focused on specific aspects of each page in the interface. Up to four questions per page were asked on each of the nine pages of the requirements analysis module. Questions were formulated so that an agreeing answer (a value greater than 3) meant the interface had met its design goals. Conversely, a disagreeing answer (a value less than 3) indicated the interface was lacking in that area. A value of 3

represented a neutral response.

The results of the trans-test questionnaire show the interface satisfied the goals set to accomplish on each page. To ascertain whether users needed help to understand the complex concepts the statement "I needed help to understand [this concept]" was presented five times throughout the survey. This question helped gauge the interface quality, as well as the difficulty of the introduced concepts. As expected, the three most complex pages received agreeing answer with an average value over 4. Evaluators had to identify a task model representing the IRC value of their system and comprehend the breakdown of that task model using Norman's stages of action. Only three out of the five introduced concepts forced most users to look for help (Figure 5).

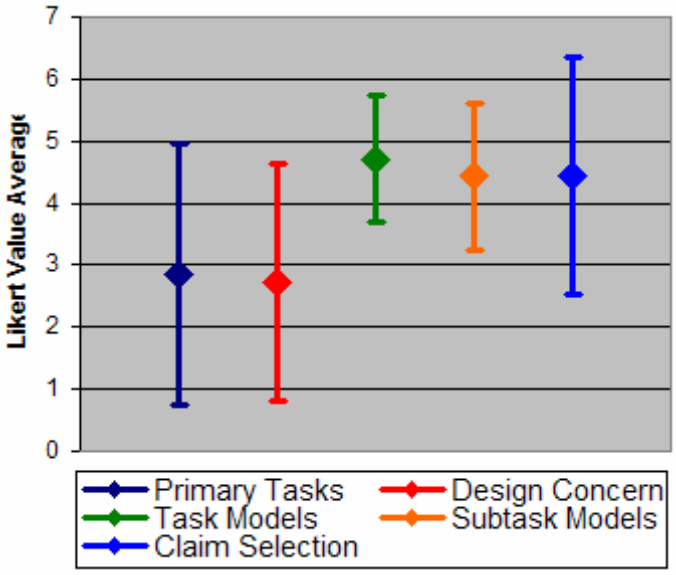


**Figure 5: Help required by users to understand concepts (average +/- standard deviation)**

When asked to evaluate the quality of the help provided on each page through the statement "I find the help link useful", testers agreed that the help was indeed helpful for understanding introduced concepts of task model and notification task selection. However, some improvements still need to be made (Figure 6).

### *5.2.2 Post-Test Survey*

After having completed the interface walkthrough and the trans-test questionnaire, participants were given a post-test questionnaire. While the trans-test survey focused on the specific design goals of each page of the interface, the post-test survey targeted the overall usability of the entire module. To evaluate the overall success of the user interaction, we relied on Nielsen's heuristics. Seven of the most critical heuristics about the interface were selected, such as help and documentation, aesthetic and minimalist design, as well as consistency and standards (Molich & Nielsen, 1990). These heuristics were embedded into questions, which asked the participants to rate the interface effectiveness. The questionnaire was set up using a Likert scale similar to the trans-test survey. The post-test survey also included three questions about the tester's ability to meet the overall goals of the interface, and three open response questions.

All of the seven heuristics obtained an average value of 4 (agree) or greater, indicating the system was well designed and that sufficient help was provided. Statements such as "appropriate reactions to the interface are obvious and intuitive" received an average of 4.43 with a standard deviation of 0.49, "the system uses informative and helpful error messages in plain English" received an average of 4.57 with a standard deviation of 1.049), while "help and documentation is focused on the right things. The explanations for terms and diagrams are satisfactory" received an average of 4.29 with a standard deviation of 1.38. These encouraging results show the interface succeeded in these areas. Although the average values for the heuristics were good, the standard deviation was large for some questions, indicating that there is still room for improvement.

Participants were asked to write what they liked, disliked, and would change about the system. Results for what testers liked confirmed some of our design decisions. However, the question about what users disliked and

would change, pinpointed some flaws in the interface design. One response revealed that "some help was not helpful, [the claims page] was confusing, [there was] too much breakaway from the simple design of previous pages. Some wordings and descriptions were confusing." Other responses included "it was smooth and easy to understand if you read the directions" and "Very robust, helpful documentation, minimalist design".

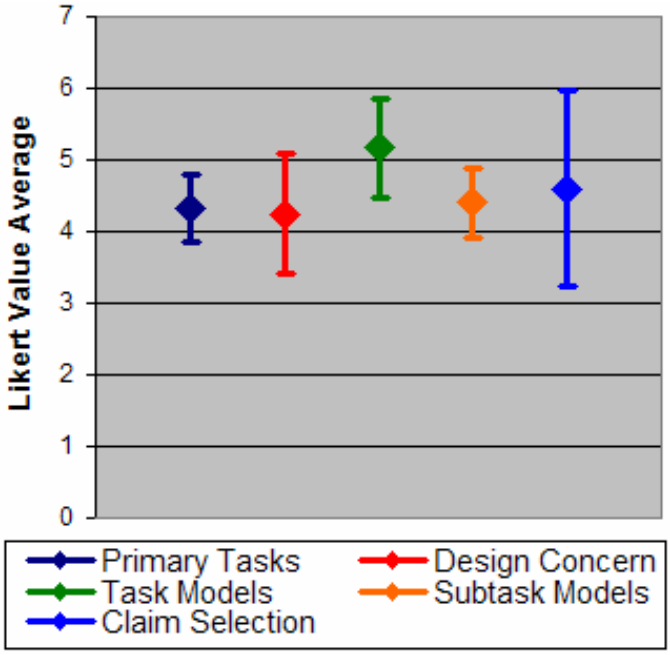


**Figure 6: Usefulness of help provided (average +/- standard deviation)**

Finally, the testers were asked three questions to evaluate their understanding of the process they had just completed. Scores were high, with the average of all being greater than 4 (agree), with low standard deviations.

## 6 Conclusions and Future Work

The results of our usability study make us confident that users were able to walkthrough the interface and understand the complex processes of conducting a hierarchical task analysis. The results show that users needed aid to figure out the interface and understand the presented concepts, but with sufficient support, users could successfully understand the complex concepts required to complete the requirements analysis module. While the current help system is good, there is room for significant improvement. In fact, users taking part of the usability study wanted a comprehensive help section gathered in a single location. This can be done by putting the help section linked from each page into a single help module. In addition, a unified help section would enable users to refer to explanations of previous encountered concepts (Figure 7).

The usage of task models within the Requirements Tool of the LINK-UP system has the potential to ensure better notification systems design. In fact, systematic usage of validated task models during the requirements analysis of scenario-based design to aid development of the design model can ensure that tasks are made clear to the designers, by leveraging an in-depth understanding and crafting of a usage scenario. It also enables the project's stakeholders to discuss design considerations, without making assumptions, by using a lingua franca. Finally, the use of task-model templates provides a method by which designers can build upon and obtain more accurate search criteria for claims, facilitating and encouraging design reuse, which tends to become more of a key aspect of design.

Currently the Requirements Tool provides task-model templates for three out of eight notification systems genre defining the problem space. For the module to be fully effective, task models will first have to be defined for the entire IRC framework. Providing task models for the entire problem domain will offer designers a starting point. Consequently, the system will have to integrate creation of new task models and support evolution of previous task models, by establishing on one hand accurate guidelines to ensure correctness, and on the other hand, standardization methods to ensure consistency among all the task models present in the system.

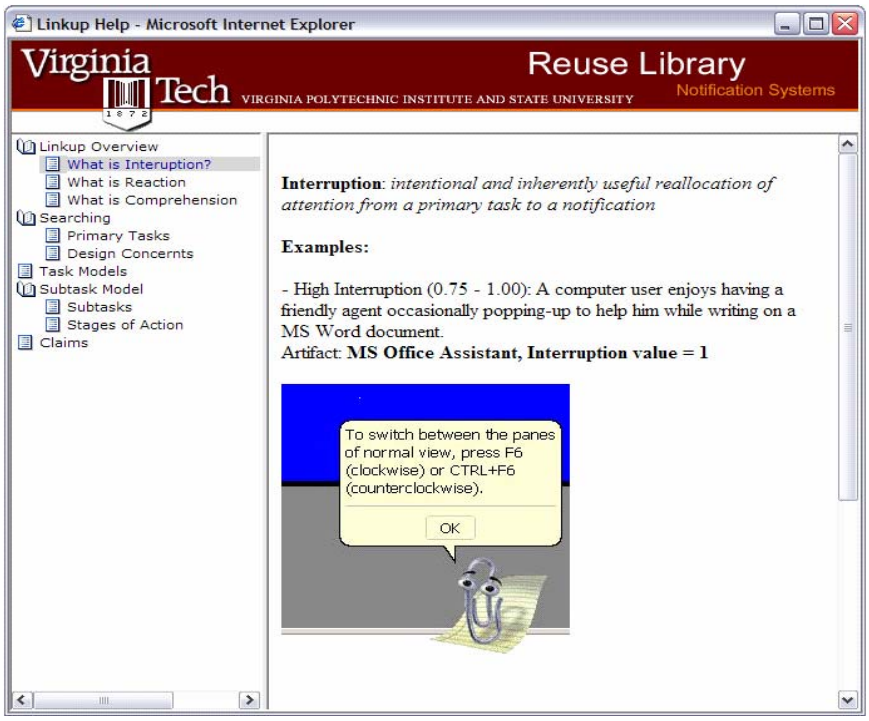


**Figure 7: Proposed help system for the LINK-UP Requirements Tool**

## 7 Acknowledgements

The authors would like to thank the Virginia Tech ASPIRES program for partially funding the effort, the students of Virginia Tech's "Design Reuse in Human-Computer Interaction" class, as well as Virginia Tech Undergraduate Research in Computer Science (VTURCS) for their assistance.

## References

1. Carroll, J. M. (2000). Making Use: Scenario-based Design of Human-Computer Interaction. Cambridge: MIT Press.
2. 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)*, 236-247.
3. 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 (DIS '04)*, 279-288.
4. Diaper, D. (2002). Scenarios and Task Analysis. *Interacting with Computers*, 14 (4), 379-395.
5. Erickson, T. (2000). Lingua Francas for Design: Sacred Places and Pattern Languages. In *Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques (DIS '00)*, 357-368.
6. 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 ACM Southeast Conference (ACMSE '04)*, 357-362.
7. Lewis, C., & Rieman, J. (1994). *Task-centered User Interface Design*. Retrieved February 8, 2005 from http://hcibib.org/tcuid.
8. 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.
9. Molich, R., Nielsen, J. (1990). In J. Nielsen (Ed.), *Ten Usability Heuristics.* Retrieved February 8, 2005 from

http://www.useit.com/papers/heuristic/heuristic_list.html

10. Mori, G, Paternò, F, & Santoro, C. (2002). CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. *IEEE Transactions on Software Engineering*, 28 (8), 797-813.
11. Nielsen, J. (1994). Heuristic Evaluation. In J. Nielsen & R. L. Mark (Eds.), *Usability Inspection Methods* (pp. 25-62). New York: John Wiley & Sons.
12. Norman, D. A (1986). Cognitive Engineering. In D. A. Norman & S. W. Draper (Eds.), *User Centered Systems Design: New Perspectives on Human-Computer Interaction* (pp. 31-61). New Jersey: Lawrence Erlbaum Associates.
13. 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)*, 362-369.
14. Rosson, M. B., & Carroll, J. M. (2002). Usability Engineering: Scenario-Based Development of Human-Computer Interaction. San Diego: Academic Press.
15. 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)*, 59-73.
16. Sutcliffe, A. (2000). On the effective use and reuse of HCI knowledge. *ACM Transaction on Computer-Human Interaction (TOCHI)*. 7 (2), 197–221.
17. Sutcliffe, A. (2002). *The Domain Theory: Patterns for Knowledge and Software Reuse*. New Jersey: Lawrence Erlbaum Associates.
18. Sutcliffe, A. G., & Carroll, J. M. (1999). Designing claims for reuse in interactive systems design. *International Journal of Human-Computer Studies*, 50 (3), 213-241.