Generalizing Interface Design Knowledge: Lessons Learned from Developing a Claims Library

Catherine Payne, C. F. Allgood, C. M. Chewar, Chuck Holbrook, and D. Scott McCrickard
Center for Human-Computer Interaction and Department of Computer Science
Virginia Polytechnic Institute and State University (Virginia Tech)
Blacksburg, VA 24061-0106 USA
{cpayne, callgood, cchewar, cholbroo, mccricks}@cs.vt.edu

Abstract

The experience of preparing interface design knowledge to be reusable allows reflection on the process, potential, and general challenges of effectively and efficiently using this knowledge in design tasks. With an interest in crafting a catalog for design claims that would implement recent reuse theory in the human-computer interaction field, we developed and implemented a unique process of creating reusable content for notification systems—interfaces used for multitasking. This process, which we describe and illustrate, extends previous work on capturing metadata related to design claims. The data and metadata are stored in the catalog, which is intended to be accessible to other designers for reuse in other domains. The multitask nature of our catalog subject matter highlights a major challenges faced in reuse: the generalization specific and contextual information (claims). The challenge of balancing abstraction with specificity to ensure both meaningful and domain-independent data is also addressed. We believe that our approach can generalize to other reuse projects that strive for cross-domain knowledge application.

Keywords: Domain Theory, human-computer interaction, scenario-based design, reuse, notification systems

1 Introduction

Imagine being able to design a software application from a catalog, ordering pre-existing parts and snapping them together—an easy and ideal approach to software engineering. But in order for this dream to be realized, someone must first *design for reuse*, collecting a wide array of components and making create mechanisms that allow others to find them and understand their use. Just as other fields have struggled to define the component structures of reusable objects, the metadata, and the search approaches, the software design community faces similar challenges.

Reuse can occur at any or all of the stages in the software and interface design process. Source code, software design, and requirements specifications are all targets for various reuse paradigms, taking the form of class libraries, object-oriented patterns (Gamma et al., 1995), or problem frames (Jackson, 2001). However, one major difference among these paradigms is where they occur in the overall system design; in other words whether they are located in the problem space or in the solution space. Reuse of software design and of source code (component-based reuse) occurs in the solution space of a design. Reuse of requirements specifications considers the problem space of a design before arriving in a solution space later in the process.

Within the early-stage software design discipline of human-computer interaction (HCI), there has been increasing research interest in understanding how to reuse design knowledge. Sutcliffe argues that the earlier reuse occurs in the development cycle, the bigger the payoff in the end. He puts forth Domain Theory (discussed thoroughly in the next section) as a method for abstracting a problem space and its related design knowledge in the form of *claims*. Claims originate from Task-Artifact Theory (Carroll, Kellogg, and Rosson, 1991), describing a particular design feature and the positive and negative aspects (upsides and downsides) of applying this feature in a certain scenario. Our work applies and extends Sutcliffe's techniques to a specific interface design challenge—notification systems—by building our own claims catalog for design reuse. This paper describes the process used in generating and generalizing the content of this library, related issues, and lessons learned from our effort.

2 Background

To situate our study within broader HCI research, we present a review of literature related to design reuse and the Domain Theory. We also introduce our knowledge reuse field of concern: notification system interface design.

2.1 Domain Theory and Claims

Sutcliffe's Domain Theory functions as a hybrid theory of expertise from cognitive and behavioral science applied

to requirements engineering and reuse. Rooted mainly in Gentner's structure-matching theory of analogy (Gentner, 1983), it states that people create loose connections across real-world problem spaces and reapply those solutions when they seem appropriate. Ultimately, Domain Theory attempts to provide a set of abstracted problem spaces to facilitate cross-domain transference of knowledge in "designer-digestible packets of reusable knowledge" (2000).

Reusable knowledge about an objects in a grounded domain exists in the form of claims, a basis of knowledge which Sutcliffe refers to as "the most powerful form of knowledge transfer that is directly contained in the Domain Theory" (Sutcliffe, 2002) and has been posited elsewhere as the chief method of empowering HCI knowledge reuse (Sutcliffe and Carroll, 1999). Perhaps the most important characteristic of claims is that they concisely state tradeoffs about a designed artifact in specific usage context according to theoretically based and empirically observed effects on a user. Figure 1 shows an example of a claim in simplest form.

Before something can be reused, it must be purposefully designed to be reusable. Sutcliffe provides a 14-point format for initially recording claims, and then a method for generalizing these claims so they can be transferred across domains (Sutcliffe and Carroll, 1999). A tension exists in this process because upsides and downsides of claims are grounded in real-world scenarios, and in generalizing a claim this context could be lost or misrepresented. Once generalized, claims can be indexed and stored in a claims library for future retrieval.

This type of reusable knowledge works well when using scenario-based design for HCI (Rosson and Carroll, 2002). Having an available set of claims provides designers with tried and tested design ideas, as well as the positive and negative impacts of using ideas. A claims catalog organized with Domain Theory should facilitate application of design knowledge across domains.

2.2 Designing Notification Systems

Our project involves creation of a claims catalog for a unique type of computer interface—notification systems. Notification systems can involve a variety of platforms and domains, delivering valued information to users while they are engaged in other activities. These systems are generally desired as a means to access valued information efficiently and effectively without introducing unwanted interruptions to primary tasks (McCrickard et al, 2003). A simple example of a notification system is a stock ticker or an email inbox status indicator. More complex systems include in-vehicle information systems, peripheral or ambient displays that slightly alter the surrounding environment, or small information summary devices that are designed to permanently reside in the corner of the desktop. These systems often use intelligent interfaces to provide recommendations

Use of Animation for Status Monitoring

{designed artifact has upside effects:}

+ Pulsing supports good peripheral detectability,
+ Users can easily distinguish new items from old,
+ Pulse rate variations can be associated with meanings,
+ Fast pulse provides a strong metaphor to urgent items,
+ Slow screen changes will not cause a lot of distraction,

{downside effects:}

– Important items may not be detected quickly,
– Multiple pulsing items are difficult to perceive at the same time and are confusing,
– Prolonged pulsing may be annoying and difficult to ignore.

Figure 1. Example claim, relating upsides and downsides user effects to designed features.

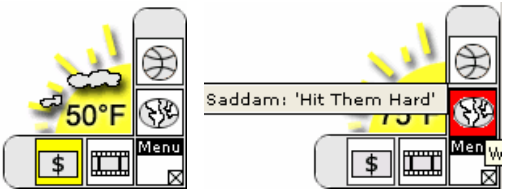

Figure 2. A typical desktop notification system, residing in the corner the desktop interface to provide convenient and subtle news alerts.

to users engaged in ongoing tasks. One such system is *In The Corner* (see Figure 2), a system which provides categorical notification of available news-related content of interest to the user. Notification systems can always be thought of as interfaces typically used with the smaller amount of divided-attention.

Effectively designing notification systems means successfully balancing attention allocation between the primary task and notification content, while simultaneously enabling access to other information (McCrickard and Chewar, 2003). In other words, it becomes extremely important to employ information presentation techniques:

- Cause the desired amount of *interruption* (enough to suitably alert the user, without undesired distraction)
- Allow a user's appropriate *reaction* to new notifications (such as understanding enough details to transition tasks or dismiss alerts)
- Facilitate long-term gains in *comprehension*, related to notification content (such as appreciating patterns and trends or linking new information to other knowledge).

Users desire different levels of these three goals depending on usage situations and context. Available information presentation options range from different types of animation (e.g., tickering, blasting, or fading techniques), use of color or audio, and other design strategies. However, using a simple rating system to describe the desirability of each parameter by users (referred to as the *IRC framework*; detailed in (McCrickard and Chewar, 2003)), we can compare notification effects articulated in claims.

3 Theory Application

Our challenge lies in organizing knowledge related to the design of notification system interfaces to be reusable and applicable for various domain requirements. How can the Domain Theory be applied to the Notification System problem domain? More specifically, how can notification systems claims be abstracted for reuse in the development of other notification systems? Our efforts involved analyzing existing notification systems to generate claims regarding their design, storing and generalizing these claims in a library, and providing a retrieval mechanism.

3.1 Content Collection

To establish our claims library for notification systems, our first step was to gather content from a wide variety of systems. We considered approximately 50 systems, to include ones that were designed within our research group, through research collaboration with other designers, and identification of systems from related literature. We intentionally selected systems that included a variety of implementation platforms and supported a range of user tasks.

Notification Collage is an example of one of the systems considered (Greenberg, 1994). This interface is intended for a large-screen display that would be situated within a common work area. The system provides a bulletin-board type forum for delivering information regarding the status of personal contacts and ongoing events. In continuing to describe our content creation process, we revisit a claim generated from this system.

As we collected content for our library, we encountered two major challenges. The first challenge stems from the tension inherent in the claim generalization process itself, trying to ensure that a claim contains meaningful design knowledge, while maintaining a level of abstraction that allows access to the claim via the library's information retrieval mechanism. Preserving this balance proved to be difficult. The second challenge was determining the precise nature and form of the abstraction layer, which provides the structure for the information retrieval process. Once resolved (as detailed in the next two subsections), we were able to develop an effective classification strategy.

3.2 Anatomy of a Claim

We began with a claims format based on the 14-point claim recommended by Sutcliffe, which in turn was based on a form developed by Sutcliffe and Carroll (1998). This format is ideal for collecting not only the important knowledge related to the theoretical grounding of the claim, but also the articulation of the design tradeoffs and context. In addition, we allow for documentation of claim history.

As we considered this format in relation to its purpose in capturing knowledge about notification systems, we realized that slight adjustments would need to be made. The more important adjustments related to the definitions of pre-existing fields. For instance, since we were accustomed to representing the effects of a notification system with an IRC rating, using the Effect (9) field seemed like the logical place to record this measurement. Similarly, Sutcliffe intended the Scope (14) field to provide relation to Domain Theory components, and we anticipated that this could be different for the notification system's primary and notification tasks. As a result, we broke this field into two subfields. These fields were originally intended to be the primary indexing mechanisms for the claims library.

We feel that the users of our knowledge repository will value the Upsides (6) and Downsides (7) of the claim, the Scenario (8) from which these tradeoffs originate, the related usage issues (10, 11), and theoretical background (12). We augmented this with the addition of the Parent Component (15) field to provide contextual reference to the system that implements the designed artifact.

Fundamental information from a maintenance perspective includes the Claim ID (1) and Author (3a) fields. Less obvious is the Relationship (13) field that further establishes the context of the claim according to other notification design issues. We anticipate that designers will be able to add most of the claim data themselves; however for quality control and index verification, we saw the need for an Editor (3b). Claims are intended to evolve as a design is iteratively developed and evaluated in actual use, so it seemed logical to include a Usage Log (16) to capture these changes. Figure 3 shows the modified claim format, indicating our changes.

3.3 Recording Our Design Knowledge

Once a claim was created in this format, our next step involved generalizing this claim to an abstract layer. Our initial definition of this layer was simply a generalized version of the original claim, similar to examples provided by Sutcliffe. Our abstract claim contained (1) primary and notification tasks, (2) IRC ratings, and (3) upsides and downsides. We planned to create a relationship between the specific claims and the abstract claims on the basis of their primary tasks, notification tasks, and IRC value. To create this relationship, we needed to describe the tasks in a way that allowed them to be grouped in different categories, which would then be captured at the abstract level.

We accomplished this by using a discrete vocabulary derived from the generic and generalized task models provided by the Domain Theory (generic and generalized tasks are listed and defined at http://www.co.umist.ac.uk/hci_design/appb.htm). The two models differ on the basis of task complexity. *Generic tasks* describe activity at a much more granular level, making them suitable to describe the notification tasks. As primary tasks tend to be broader in the range of activity that they encompass, the

Field	Description
1. Claim ID	Unique identifier for the claim
2. Title	Description given by the claims author
3a. Author	Researcher(s) or practitioner(s) who developed the original claim
3b. Editor	Person who created the claim record for the library and established the IRC rating
4. Artifact (Design Abstraction)	Description of the claim artifact in terms of the design abstraction
5. Description	Natural-language description of the claim
6. Upsides	Positive effect of using claims on a management objective of system effectiveness
7. Downsides	Negative effects of using claims
8. Scenarios	Originating scenario in which the claim was derived
9. Effect (IRC Rating)	Desired, measurable effect that the implemented claim should achieve
10. Dependencies	Problems that need to be solved before getting to the root issue
11. Issues	Management issues possible influenced by the claim
12. Theory	Underlying theory explicitly reference by the claim
13. Relationship	Interclaim links that describe how claims evolved during the history of the investigation
14a. Scope (Primary Task)	Generalized Task keyword description
14b. Scope (Notification Task)	Generic Task keyword description
15. Parent Component	System name and information
16. Usage Log	Running commentary on experience in using the claim

Figure 3. Modified Claim Format where the first 14 fields correspond largely to Sutcliffe (2002).

more complex *generalized tasks* (compositions of multiple generic tasks) were used. Figure 4 shows how the claims creation process mirrors the design process.

3.3.1 Defining the System and Usage Situation

The first step was to identify a system from which we wanted to capture usability experience, like Notification Collage. Next, we identified the usage situation in which we wanted to evaluate the system. Here we decided to look at the live video feed capability of Notification Collage in the case where the feed supports monitoring the status of another person. However, the key notification system design challenge would be that this monitoring would occur while the user is simultaneously engaged in other office work. Next, we identified the general design technique (Design Abstractions in Figure 4) that characterizes how the system or system component interacts with the user. Discussion of this characteristic is reserved for the next subsection. Of interest here, we also identified a claim title and chose the key word(s) to describe the primary and notification tasks, beginning the development of a claim record (see Figure 5).

In this instance, we chose monitor, interpret, and evaluate as notification tasks. The user has goals of maintaining awareness about the availability status of another person (monitor), and based on his/her understanding of this status (interpret), determines a suitable time to interact (evaluate). We chose planning-scheduling as the primary task because the user will incorporate their understanding of the other person's availability to sequence collaborative and non-collaborative work activities.

3.3.2 Developing the Scenarios and Tradeoffs

Next we develop the scenario, which is used to ground the usage context of the system (Rosson and Carroll, 2002). To continue the example with Notification Collage, we created a scenario to describe a typical usage situation. From this scenario, we identified the pros and cons of the design, which are articulated in the upsides and downsides for the claim. The description summarizes the designed artifact in the context of use, as shown in Figure 6.

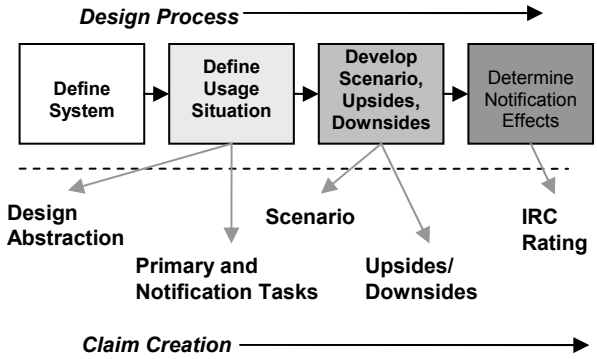

Figure 4. The process we followed to establish claim content, describing design knowledge of a notification system artifact.

2. Title	Video Feed for Status Monitoring
4. Artifact (Design Abstraction)	Video
14a. Scope (Primary Task)	Planning-Scheduling
14b. Scope (Notification Task)	Monitor, Interpret, Evaluate
15. Parent Component	Notification Collage

Figure 5. A portion of the claim record for our Notification Collage example, resulting from analysis of the usage situation.

3.3.3 Determining Notification Effects

Having established the review of the design feature in the context of its scenario of use, we turn our attention to the notification effects. This is the primary knowledge that notification systems designers will find valuable for new design requirements. The IRC rating (discussed in Section 2.2) provides a measurable description of an artifact's impact on a user. After establishing this rating, we make note of any relevant dependencies, issues, or theories that should be considered during notification system design.

Figure 7 continues our example, showing the completion of all non-trivial claim fields.

9. Effect (IRC Rating)	Interruption = 0.2, Reaction = 1.0, Comprehension = 0.9
10. Dependencies	Transmission bandwidth, resolution. The refresh rate (presumed at every 5 min, for some privacy preservation and lower bandwidth transmission) prevents monitoring of the display during natural primary task break points.
11. Issues	Privacy of monitored person
12. Theory	Ackerman's social-technical gap (privacy); McFarlane's user-initiated interruption (refresh rate); Dourish & Greenberg (awareness for reciprocity and privacy)

Figure 7. Claim fields relating to consideration of notification effects, pertaining to the Notification Collage example.

3.4 Cataloging the Claims

During this claims creation process, we indirectly established a classification system or indexing mechanism for the catalog. However, this process deviated considerably from the method Sutcliffe proposes. This section describes our experiences, which ultimately lead to our recommendations regarding the design of a claims library.

We initially encountered a problem with articulating the primary task for our usage cases. Some of our systems were very ubiquitous, which made identifying a primary task in the general case practically impossible. This in turn meant that we could not selectively map the specific claims to the abstract claims, as Sutcliffe and Carroll's factoring process intends (1999). We solved this problem by not focusing on what the notification system can support, rather on a specific primary task goal. This direction allowed us to be able to identify a discrete primary task.

5. Description	Using an extraneous video feed displayed on a large screen display to monitor the presence of a remote individual.
6. Upsides	+ Live video allows a quick and easy way of showing presence, + Posting of live video, sticky notes, slide shows, etc, affords a wide variety of media forms, + Lack of audio decreases interruption and information overload, avoiding sensory overload, + Customizing the rate at which the video feed updates allows the user to control interruption.
7. Downsides	– Live video broadcast reduces privacy for users, – User-initiated interruption is hindered because users don't have the ability to control the refresh rate on a video feed, – The user can miss a change of status because of lack of audio or other non-visual cues.
8. Scenarios	*Bob is working with Alice on some paperwork in their lab. Bob must frequently go over to Alice's workstation to get her to look at the paperwork and sign documents that are needed. Alice, however, is often not at her workstation. Bob can fire up the Notification Collage (NC) on a second monitor and ask Alice to post a video feed of her workstation area. Bob will be able to continue doing his work, but he is aware of Alice's presence. By glancing at the NC when Bob takes a break or has a need to talk to Alice, he will be able to quickly realize if Alice is present at her workstation. He will no longer waste time walking across the lab to find Alice.*

Figure 6. Reusable design knowledge—sample claim description, scenario, upsides, downsides.

By increasing our dependence on the scenario, however, we raised a new issue with respect to abstraction. Our claim became even more reliant on context, which is contrary to the principle of abstraction and basically removed the property of domain extensibility from our abstract claims. To resolve this dilemma, we decided to change our original definition of an abstract claim. Rather than having an abstract claim exist as a generic, higher-level variation of a specific claim, we decided to focus on functionality.

Instead of abstract claims, we used a "footprint" approach, recognizing that each specific claim exists in an abstract, four-dimensional space based on its primary and notification tasks (Scope) and two new indices: IRC ratings (Effect) and Design Abstraction (Artifact). Because the primary and notification tasks were based on the generic and generalized task models described in the Domain Theory, they already placed the claim on an abstract level. The Design Abstraction attribute is a generalized way of describing how a system interacts with a user (e.g. color, animation, or audio). IRC ratings describe the effects that a system will have on the user (e.g. causing high levels of interruption) in non-specific terms. A claim's existence in this four-dimensional space constitutes library abstraction.

Continuing our Notification Collage example, we look at the four variables used to place the claim in the library. Since the primary and notification tasks were already in the vocabulary of the tasking models, they are ready to be mapped, as is the IRC. The final variable needed to complete the abstraction of our claim is the Design Abstraction

variable. To find this, we considered possible Design Abstractions and chose those that best described the component of the system we were evaluating. In this case, we chose Video as our design abstraction. Once we had identified these four variables, the claim could then be included in the library. Note that in this case, there are three notification tasks, so there will be three different combinations of these variables. The union of all combinations is the footprint. If a user's query falls anywhere in that footprint, the claim is returned.

Deviating from the original abstract claim design came with a loss, however. While the footprint serves as an effective abstraction layer for information retrieval, the holistic effect of these four dimensions does not relate any sort of meaningful design information. Were we to use the abstract claim format based on the specific claims (as proposed by Sutcliffe and Carroll (1999)), we would be able to capture a general, context-free design claim that would work regardless of domain. To maximize the benefit of a catalog of design knowledge, cross-domain applicability is highly desirable. We revisit the techniques that we used to alleviate this concern in Section 6.

4 Intended Claims Catalog Usage

Having demonstrated the application of claims creation and classification for notification systems, we now describe how our claims catalog could be used within a typical scenario-based design process.

4.1 The Essence of Design Knowledge Reuse

Once implemented and filled with claims, the library would offer designers a convenient location from which to draw knowledge when conceiving new systems. From a scenario-based design (SBD) perspective, this could be used to generate, reinforce, or provide different viewpoints on claims related to the system being developed. This could be achieved with the library in a number of different ways. One way would occur when the narrative of use is created. This scenario describes the user's goals and the processes that take place when the system is being used. By accessing library content, designers could be prompted to include ideas in scenario-writing that they otherwise would not have considered. From this narrative, one could generate claims that may highlight a particular aspect of the system or show a requirements constraint.

Without the library, a claim would have to be formulated from the designer's existing knowledge and hypothesized from the obvious implications derived from the scenario. With the library, the designer could search for similar systems by selecting the appropriate search criteria to retrieve claims made about systems with similar abstractions. Typically, this would include knowledge applicable to the current design task. This knowledge would contain the upsides and downsides of claims with regard to issues noted in previous and current design scenarios. This form of reuse could prove invaluable.

4.2 Cross-Domain Reuse

While the claims within the library are derived solely from notification systems, they can be applied to any domain. The information retrieval mechanism (see Figure 8) facilitates searching regardless of domain context or claim details, preserving this information within the claim to allow the designer to determine its relevance.

For example, large screen displays and PDAs would seem to share little in common, aside from the fact that they can both be used as notification systems. However, in notification tasks, users may share many of the same goals (e.g. allowing a user to learn of a change in system state with little interruption). Claims made for one system may not have exactly the same effect as they would on the other, but the general deign advice might be similar. For example, a flashing screen may not be as noticeable on a PDA as it would be on a large screen display, but it would certainly have the effect of notifying the user of an event, regardless of platform. Using the library to reveal such claims would help in a designer's brainstorming process–providing insight about origins of the claim, while allowing the designer to determine applicable information.

Figure 8. Screenshot of information retrieval interface for the notification system claims library.

4.3 Retrieving Claims

Imagine that a designer is developing a PDA application to monitor the status of complicated factory equipment that has a fairly high failure rate. After a scenario of use is created (such as the one in

Figure 6), the design effort focuses on creating a system that meets the user goals highlighted in the scenario; specifically a system that supports the notification task of monitoring, the Design Abstraction of feedback, high reaction and interruption, and most types of primary tasks. At

this point, the designer could brainstorm possible ways to implement the system for the scenario. To expedite thinking of different designs for the system and identifying the costs and benefits of each, the designer turns to the library. There, he is able to retrieve claims on systems with similar goals, along with the pros and cons of each implementation.

A number of different claims are likely to be returned, a few of which the designer is immediately able to rule out as being unrelated. Others offer advice on obvious implementations, but give complete descriptions and tradeoffs about using them. As an example—*using text messages for monitoring offers low interruption and high comprehension*—gives sound and tested advice. Near the end of the search results is a claim that intrigues the designer. It is the example claim previously described from the Notification Collage system—"Video Feed for Status Monitoring." Although this claim does not fit exactly within the scenario as described above, the designer now sees the possibility of using video within the new system to achieve even better results. Instead of using text messages to describe the possible problems with the equipment, the designer now envisions the system taking a picture of the faulty machine that would give the user very detailed information about the problem and possibly its cause.

With the knowledge gained from browsing the library, the designer is able to revisit the initial scenario and create a new one using the video feed. The new design demonstrates an even more powerful system that, in many cases, would save the user from having to physically go to and inspect the machine to learn of the problem cause. When the designer is finally content with the design, he is able to download the executable components linked to the claims found in the library and use them to construct his own prototype. After implementing the designed system, the designer adds his own specific claims to the library and updates the claims referenced during the design with links to the new ones gained through his experience.

5 Initial Feedback & Future Work

After developing the system architecture and claims catalog user interface, we were eager to receive feedback about how well it supports the design for reuse process.

5.1 Initial User Evaluation

We wanted to get feedback on how well our abstraction method supported information retrieval, so we had nine participants use the system by trying to apply the populated library in a series of design tasks. While the design tasks tested many aspects of the entire library, we focused on how useful the participants felt the claims were in their design process and how intuitive and meaningful the retrieval process was. We were primarily interested to see if library users were able to negotiate the retrieval process through the generalized and generic tasks, after abstracting the given design tasks.

5.1.1 Methodology

The way the participants interacted with the library was as follows. A design problem was presented and the participant was asked to search the library for claims that would help them most with the task. They would then create a query based on a combination of primary tasks, notification tasks and Design Abstraction key words. The key words were listed on dropdown lists and used the same vocabulary that was used in describing the primary and notification tasks and Design Abstractions. Based on the query, a number of claim summaries were returned. If the participants wanted more information on a claim, he could expand it to view the entire claim. Participants would select the claim if it was determined to be helpful to the design task.

5.1.2 Results and Feedback

After the participants finished the series of design tasks, we asked them for feedback on whether the tool helped them locate relevant design knowledge, how effective the library was in doing so and what the library was most useful for. Roughly half felt the library was useful, and the majority overall felt it was most useful in inspiring new ideas over reinforcing design assumptions. With respect to how effective the library was in helping out with the design task, four said it was effective for all scenarios, four said it was effective to some degree in all scenarios and one said it was effective for some scenarios. Based on these comments, we feel our claims content and format was successful in terms of storing design data.

We also received feedback on our key words, which formed the basis of abstraction. While the majority of the participants felt they were able to match the key words to the given scenario, some felt that the key words were too vague and that there were too many of them. The complaint on vagueness is a testament to one of the difficulties in finding the right balance of generality and specificity with respect to abstraction.

5.2 Future Work

Future work will continue to probe the design for reuse aspect of our claims library. We would like to extend our current work in three ways: formal usability testing, content development, and user interface refinement.

From a testing standpoint, we would like to conduct usability tests to determine which of our four abstraction variables are the most meaningful in queries. To do this, we plan on collecting data use of the database to find claims to support a design task. During these tests, we would evalu-

ate the effectiveness each variable has in queries. We also plan on gathering general usage statistics as well.

In order to support a robust testing regime and long-term use, we also plan on developing more content. Not only does more content allow a viable database, we gain more experience and learn more things about design for reuse process, which can be incorporated into our effort.

Finally, because information crosses domains on the basis of users interacting with claims from different domains, the user interface is important. In particular, we would like to investigate different ways to conduct the queries based on user feedback about the interface, as well as lessons learned about the role each variable plays in retrieval. As a general goal, we recognize great benefits in developing a recommender system that can facilitate specification of search criteria and suggest content that may be useful.

6 Discussion and Conclusions

Advocates of reuse tout it as efficient, both from a knowledge management standpoint as well as cost. *Design by reuse*, however, can only be implemented after the *design for reuse* process has created enough reusable material. The primary cost of reuse resides in the design for reuse effort. The question we then ask is: how much additional burden does design for reuse place on the designer?

Take the professional scenario-based designer, designing for reuse. As she steps through the development process, we can see that she has the additional overhead of having to prepare her claims for future reuse. Depending on how many other people in her organization are also taking on the extra responsibility for creating reusable components, she might never directly benefit from her effort, at least not in the short-term. The disparate distribution of costs and benefits, which is a common theme in computer-supported cooperative work as a whole (Grudin, 1994), is a major factor that reuse must overcome to be effectively implemented. Moreover, not only does the design for reuse process put additional burden on the designer, it requires additional documentation through the product's lifecycle, since new claims or revisions to existing ones will continue to be made as the product is developed and implemented.

The major challenge we faced in our design for reuse effort was abstracting relationships between system dynamics. In our case, to generalize the dual nature of the usage scenario, we had to abstract the notification and primary tasks in conjunction. We were unable to do so explicitly because we kept losing the relationship between the two tasks. One reason for this is the tasks were abstracted to different levels (notification tasks were described more granularly). When both tasks were then further abstracted, one of the variables would dominate the other, and the original relationship between the two would be lost.

We were able to overcome the problem of explicitly defining an abstract claim by indirectly doing so. We created a footprint for the claim, through the combination of four indexing dimensions. This technique provided an abstract layer that could support retrieval. Unlike the original abstract claim idea, however, the footprint does not provide abstracted meaning, losing a second source of domain-independent claims.

Our original goal was to implement a claims library that supported domain-independent knowledge transfer. We believe we have been successful in doing this, and recognize a need to express the quantifiable benefits of design for and by reuse. The designer's ability to interact with the different domain-based claims that are returned and to integrate them into her process is proof that this system implements the spirit of Domain Theory.

References

[1] Anderson, R.E. (1992) Social impacts of computing: Codes of professional ethics. *Social Science Computing Review*, 2 (Winter 1992), pp. 453-469.

[2] Carroll, J.M., Kellogg, W.A. & Rosson, M.B. (1991) The task-artifact cycle. *Designing Interaction: Psychology at the Human-Computer Interface*. Cambridge Press. 1991, pp. 74-102.

[3] Conger, S., and Loch, K.D. (1995) Ethics and computer use. *Communications of the ACM 38*, 12 (December 1995) pp. 30-32.

[4] Greenberg, S. & Marwood, D. (1994). Real time groupware as a distributed system: Concurrency control and its effect on the interface. In *Proceedings of CSCW'94*, ACM Press, pp. 207-217.

[5] Grudin, J. (1994). Groupware and social dynamics: Eight challenges for developers. *Communications of the ACM*, 37(1), pp. 92-105.

[6] Jackson, M. (2001). *Problem Frames: Analyzing and Structuring Software Development Problems*. Harlow: Pearson Educ.

[7] Mackay, W.E. (1995) Ethics, lies and videotape... In *Proceedings of CHI'95*, Denver, CO, ACM Press, pp. 138-145.

[8] McCrickard, D. S. and Chewar, C. M. (2003). Attentive user interfaces: Attuning notification design to user goals and attention costs. *Communications of the ACM 46*, 3, ACM Press, pp. 67-72.

[9] McCrickard, D. S., Czerwinski, M., and Bartram, L. (2003) Introduction: Design and evaluation of notification user interfaces. *International Journal of Human-Computer Studies 8*, 5 (May 2003), pp. 509-514.

[10] Rosson, M. B. and Carroll, J. M. (2002). *Usability Engineering: Scenario-based Development of Human-Computer Interaction*, Morgan Kaufmann.

[11] Schwartz, M., and Task Force on Bias-Free Language (1995). Guidelines for Bias-Free Writing. Indiana University Press, Bloomington IN.

[12] Sutcliffe, A. (2000). On the effective use and reuse of HCI knowledge. *ACM Transactions on Computer-Human Interaction 7*, 2, ACM Press, pp. 197-221.

[13] Sutcliffe, A. (2002). *The Domain Theory: Patterns for Knowledge and Software Reuse*, Erlbaum Assoc.

[14] Sutcliffe, A. G. and Carroll, J. M. (1999). Designing claims for reuse in interactive systems design. *International Journal of Human-Computer Studies, 50*, 3, Academic Press, pp. 213-241.