Links for a Human-Centered Science of Design: Integrated Design Knowledge Environments for a Software Development Process

C. M. Chewar[1] and D. Scott McCrickard
Center for Human-Computer Interaction and Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0106
{cchewar, mccricks}@cs.vt.edu

Abstract

Based on extensive empirical observation of design activities that might be supported by a knowledge repository, we report conclusions from three case studies. Seeking to improve research infrastructure necessary to cultivate a "science of design" within human-computer interaction, we focus on identifying essential activities that help proceduralize the key requirements of knowledge management within a software development effort. From related literature, we selected five focus points for our analyses, which in turn, guided development of our repository in terms of how design knowledge is used, reused, and harvested through system tools. The case studies successively validate potential activities, while exposing breakdowns in process or practice that show promise of being resolved with additional tool features highlighted in other cases. Emerging largely from our case studies, we present general guidelines and tradeoffs for developing a design knowledge repository, as well as directions for further empirical study.

1. Introduction

As our field becomes more mature and we settle on processes, the prospect of having reusable knowledge units has much appeal. Within the concerns of software development, human computer interaction (HCI) researchers and engineers ensure that interfaces allow users to accomplish goals. HCI as an area of research concerns itself with basic questions of human perception (e.g., choice of colors, and elemental interface layout), ranging to interaction techniques (e.g., design of widgets, and manipulation strategies) to selection of critical task-related activities. The processes essential to human-centric design have only begun to take shape in the last two decades, yet still are inconsistently applied and are continually evolving with the introduction of new technologies. However, researchers have already begun laying the groundwork in establishment of a "science of design" to allow a more systematic, deductive approach to advancing our body of knowledge.

We expect that three key benefits will emerge from establishing infrastructure that will support a science of design. First, we need to facilitate the transfer of knowledge from basic researchers (i.e., psychologists, sociologists, industrial engineers) to software developers, who combine the guidelines and principles developed through basic research into a working software system. In turn, the use of this system provides opportunities for reflection on the basic theories, which should transfer back to appropriate research activities. Second, we wish to have more thorough understanding of the design spaces we hope to advance. This understanding must convey to a community of researchers the common problems where solutions exist, as well as those that still require innovative design initiatives—thus, allowing comparison of systems, suggestions for reuse, and recognition of progress. Third, we see the benefits that emerge from all well-established science disciplines: a practical, value-adding engineering process. That is, through the structures on which we base our science, we should also be able to develop new interface products reliably, cost-efficiently, and of higher quality.

One approach toward developing a science of design infrastructure has been the development of knowledge repositories to hold potentially reusable techniques for achieving usability goals. There have been several approaches underlying various implementations of design knowledge repositories (as reviewed in the next section). However, a common theme seems to be that if we can manage this knowledge in a more efficient way, researchers will be able to apply it between domains, sparking cross-domain reuse and innovation.

Our work seeks to extend the conceptualization of design knowledge repositories within the discipline of HCI and the practice of software development. We wish to adopt work on knowledge management and support for design reuse from other disciplines, where it has been more completely applied and understood. As we set out on this goal, we recognize three fundamental questions that must be answered:

[1] Author's current address: Department of Electrical Engineering and Computer Science, United States Military Academy, West Point, NY 10996

1. What are the most valuable paradigms from knowledge management and software reuse that will support the activities required of a design knowledge repository?
2. How can we embody foundational HCI concepts in the general design of a knowledge repository?
3. How might a design of knowledge repository activities accommodate novices in a domain or novice designers?

To probe these questions and provide general recommendations for design knowledge repository development, we reflect on a specific design domain—notification systems—and three interface design cases that benefited from design reuse.

2. Related work

In this section, we begin our review broadly by considering general knowledge reuse supporting innovation, then turn our focus more specifically to software development and HCI. Through this, the difference in scope between knowledge management and software reuse becomes clear, and we identify key points to focus the work described in this paper. We also review background related to a specific kind of design knowledge repository, a claims library.

2.1. Knowledge management and reuse

We first look to work that has emerged from the management science community to better understand the general knowledge reuse process. Majchrzak provides a study serving two purposes to this work. First, their case-centric approach proved to be an effective research method for examining existing processes and supporting key findings related to a knowledge management approach [9]. As this paper presents details about their research design and case selection process, we are able to follow a similar method in our own domain. Second, they motivate the need to build an argument distinguishing between knowledge transfer for replication and for innovation. Through their own review of literature in the field, they argue that support for radical innovation must be approached much differently in terms of staged processes. Their conclusions present six distinct stages in a knowledge reuse for innovation process that take researchers through the processes of reconceptualizing a problem, deciding to search for reusable ideas, conducting the search, evaluating potentially reusable ideas (both briefly and in-depth), and developing the innovative idea. Their recommendations are unique to the specific knowledge reuse goal—that of support for innovation—suggesting the importance of tailoring design of reuse-related activities to specific goals. Within their staged model, they emphasize the importance of providing search facilities at multiple levels of detail and emphasize the importance that innovators place on meta-knowledge of knowledge units.

As we move closer to the domain of software development, Rus and Lindvall [15] introduce the role and issues pertaining to knowledge management in software engineering. From this work, the distinction between knowledge management and reuse becomes clear. Knowledge management involves the capture and dissemination of an individual's experience, to include the background, assumptions, and decision criteria related to a particular fact or decision, policy or practice, or other type of knowledge. The authors highlight many motivations for knowledge management within software engineering, to include business outcomes like decreasing production time and increasing quality, and satisfying information needs within an organization (e.g., providing knowledge about new technologies, specific domains, or collaboration opportunities). At a lower level, software reuse involves the core code components developed by programmers and placed in a common repository. While knowledge management may include support for software reuse, a more full knowledge package may include information related to the project budget, schedule, stakeholder requirements, and other background needed to understand programming approaches.

Specific to software reuse, Krueger provides a survey of approaches to software reuse and develops four key dimensions a reuse system must support: abstraction, selection, specification, and integration 0. This emphasis on abstraction and generalization resonates with Majchrzak's recognition of the importance of a multi-level search strategy for untargeted browsing. Within the requirements engineering and HCI fields, Sutcliffe has developed a similar argument for supporting reusable design knowledge components, also focused on techniques for facilitating abstraction and generalization [16]. He argues that, for collection and maintenance of reusable components to be economically viable, they must have some potential to apply to a broad range of new problems within multiple domains. Therefore, Sutcliffe has designed a knowledge representation ontology for generic user tasks and information exchange operations—all conveyed as abstract models. With his general views of problems, developers are provided with a structure to describe context for processing requirements. This structure can be applied to specific claims about the psychological effects of design features, allowing such claims to be applied in similar context that may be characteristic of a different domain.

2.2. Knowledge units for HCI reuse

Carroll and others have long argued that the emergence of a design science within the practice of interface development would result from analysis of design rationale. Carroll first introduced the notion of *claim schemas* to summarize key aspects of design rationale, expressing the psychological effects on users that result from a designed feature within a particular scenario of use [2,3]. Claims express both positive and negative tradeoffs that may be expected, providing a record for others to understand why specific design decisions were made. Carroll and Sutcliffe have demonstrated how claims can be combined and reused in other design contexts to result in design transfer and innovation [17]. Together, they develop a vision for *claims libraries*, design knowledge repositories consisting of general versions of claims, which designers browse to aid their development.

Certainly, claims are not the only approach. For example, compelling arguments have been made for design patterns as a lingua franca for HCI knowledge storage [6]. While other approaches may provide more specific information related to the actual design solution, claims provide more direct focus on the psychological effects at the core of requirements-level HCI focus (where we expect reuse to be most beneficial).

2.3. Key focus points for repository activities

As we reflect on some of the arguments from the related work, particularly with respect to the fundamental questions raised in the introduction, we can begin to identify five key focus points for developing *activities* for a design knowledge repository (i.e., broad tasks that a user would accomplish with a system):

1. Design knowledge repository activities need to be developed to *support designers' specific motivation to reuse*. For instance, Majchrzak distinguished between reuse for cross-domain acquisition (as Sutcliffe targets) and reuse for innovation.

2. Processes that allow designers to *narrow focus of concern to identify components of interest* are critical activities in design knowledge repositories. For example, Majchzak describes a three-level search strategy that helps developers move from broad to detailed considerations of their requirements.

3. Designers need to be able to *index components in abstract terms,* allowing application to a broader range of requirements. Both Krueger and Sutcliffe note the central role that abstraction plays in reusable component design—that by stripping the context from the initial presentation of content, the general underlying solution can be recognized by future developers.

4. To allow developers to relate to generic abstractions, use of design knowledge repository structures must *provide sufficient context.* Through Sutcliffe's use of generic building blocks (object models and task structures), he demonstrates a structure that allows designers to judge details of the original context, helping them match it to their own. Though seemingly contradictory to the previous point, context, while not initially as important as other factors, in fact is important at progressive levels of component evaluation.

5. Repository solutions must also *account for the many barriers to knowledge management.* Though not described in our related work, Rus outlines several challenges, such as the overhead in preparing components to be reusable and the fast pace of technology change.

These key focus points, revisited in Table 1, become central to our analysis of design practice and recommendations for an approach to developing design knowledge repositories.

3. Creating a design knowledge repository

Having synthesized from literature key focus points for development of activities in a design knowledge repository, we turn to a specific design area to explore concrete requirements and implementations of these activities. Although we focus on a specific design domain to achieve depth in our analysis, we continue to probe activities that can be generalized to any human-centered design concern. First, we introduce our specific design domain—notification systems. Next, we discuss some initial decisions and continuing questions driven by the key focus points. These questions motivate the case studies presented in Section 4, which further illustrate our key focus points and proposed guidelines.

3.1. Designing notification systems

In selecting a specific design area to support with a design knowledge repository, we had two criteria. First, we wanted to ensure that design challenges inherent in the area were deep and primarily based on human psychological characteristics, rather than technological constraints. Second, we were eager to identify an area of design research where advancements could inform a variety of other design challenges (e.g., support for universal access, ensuring usability for different user demographics, and adapting information display with situationally appropriate presentation) and design domains (e.g., ubiquitous computing, computer supported collaborative work (CSCW), and multimodal interfaces).

We argue that the emerging discipline of notification systems provides an ideal design area of inquiry. Specifically concerned with interfaces that are intended to be used in divided attention situations, *notification* *systems* deliver additional information of interest to their multitasking users without introducing unwanted interruption to a primary task [11]. These interfaces can be found in many implementation forms and on a variety of platforms. Perhaps classic desktop systems are the most readily identifiable—instant messengers, status programs, and stock tickers. Other familiar examples hint at the range of potential systems, such as in-vehicle information systems, ambient media, collaboration tools, and multi-monitor displays. Since notification systems are often lightweight tools (often small peripheral displays in the corner of a computer desktop) informing users about everyday information (e.g. airline ticket prices, news headlines, presence of collaborators or loved ones), designers are generally able to address important concerns with a relatively simple implementation effort.

However, a user's initial acceptance and continued use of a notification system largely depends on satisfaction of their multitasking usage goals—leading to difficult design tradeoffs that are rooted in human psychological constraints. In an effort to understand the essential design questions within the area of notification systems, we have conducted extensive work that has led to identification of general sub-classes of systems and information processing characteristics of the sub-classes. This in turn has led to key user goals underlying divergent design requirements—control of interruption, reaction, and comprehension outcomes (or *IRC*) as users trade their attentional resources for some anticipated utility from a notification system [10]. Other work has formalized the IRC factors so that they can be used as a scoring mechanism within interface usability testing practice, and the IRC factors have also proven suitable for classification of specific design artifacts [4]. That is, we have tools available that help classify a design artifact in terms of the relative amount of user interruption, reaction, and comprehension it affects (an *IRC rating*).

3.2. A claims library for notification design

As we begin to conceptualize what a claims library *might be like* for notification design artifacts, it is important to recall our general goal of promoting a science of design through transfer of basic research results, understanding of design spaces, and facilitation of a practical software interface development process. With this goal in mind, we revisit the key focus points introduced in the previous section. Although we are able to make some initial decisions toward a specific approach for a design knowledge repository based on the design domain (notification systems) [14] and general arguments that guide HCI research and practice, other focus points only generate questions for further reflection at this point.

Addressing the first focus point, since we need to be able to *support designers' specific motivations to reuse*, we must clearly identify those motivations. Of course, we wish to support design-related innovation, so we look for the answer to a narrower question: *Outside of a specific design project, how might we gauge opportunities where innovation is needed and judge the merits of a proposed solution?* Norman's theory of action has long been thought of as a guiding beacon for HCI (i.e., as seen in [1]). As part of this theory, Norman recognizes two different expressions of a task (physical and psychological) that must be resolved within a human-computer interaction system for it to function as expected [13]. Inherent in this resolution process is achieving consistency between two conceptual models—the *design model* held by the designer and the *user's model* based on the user's understanding of the system. Reuse for user-centered design should further our ability to resolve these models (judging the merits of a proposed solution), or identify general problems that are not addressed by sufficient solution options (suggesting the opportunity for innovation). Therefore, our design knowledge repository will help designers compare the two conceptual models for user requirements and design artifacts in question.

Key focus point #2 involves development of design knowledge repository activities to *narrow the focus of a designer's concern to identify components of interest*. In HCI, the components of interest should have impact on psychological effects, and for notification systems design, we have argued that the IRC factors are the critical psychological effects of interest. In more general terms, these three psychological effects are called *critical parameters*, a term introduced to the HCI design community by Newman. To conduct meaningful modeling and usability evaluations that allow systems to become progressively better, Newman argued we first must define or adopt critical parameters, or figures of merit that transcend specific applications and focus on the broader purpose of the technology [12]. Rather than develop a design knowledge repository that includes reusable knowledge related to a wide variety of psychological effects, we propose a claims library that focuses on most essential critical parameters of a system class—for notification systems, the IRC factors.

For key focus points #3 and #4, Sutcliffe's Domain Theory, coupled with IRC ratings, provides a starting point for *indexing* and *providing context* for components. However, we are uncertain how these types of generic structures will be used in the course of design work, so further reflection is needed. Additionally, we need to more carefully study typical design processes to overcome *barriers to knowledge management* (the fifth focus point) that may be common in design practice.

4. Case Studies

In this section, we present three case studies of actual notification systems design projects, all of which include some component of design artifact reuse supported by a prototype notification claims library. Each case depicts real events and outcomes, as experienced by other designers within our research lab. Case observations were made about once per week on average, including discussion with designers, review of materials and testing results, and reflection on final reports. Throughout a narrative of each design process, we discuss implications for activity development within the claims library. Case observations expose breakdowns in proposed activities, suggest alternate implementations, and validate specific parts of our ongoing development effort.

4.1. Case 1: Thumb-Type

The first case describes a development process for an input method to an in-vehicle information system (for additional details see [1]). The intent of this design was to provide a competing technique to input methods like voice recognition or gesture systems to be used in the attentionally demanding driving situation. Although product development was discontinued after initial user testing, we view this as a successful design case because access to reusable knowledge helped expose design flaws before the development effort became costly.

Having heard that a colleague successfully developed a graffiti-like input method for in-vehicle digital music selection [7], the designer was eager to adapt the input method to allow selection of characters for a vehicle navigation system. The prior development effort for the graffiti-like system had involved several months of hardware and software development, as well as lab-based user testing. Through a comparison with a touch-screen interface and a voice recognition mockup, the graffiti-like input method was shown to promote enhanced awareness required for complex menu traversal, thus decreasing distraction from the driving task. While the graffiti-like method required fairly high learn time, as an input method for digital music selection that would be generally marketed to a young demographic eager to adopt new technology, this was not determined to be a serious issue.

These psychological effects and the design decisions that caused them were recorded as claims within a prototype version of our claims library. Each claim was characterized by an IRC rating describing the usability test outcomes as well as the generic task(s) it supported (based on Sutcliffe's Domain Theory [16]).

As he began reasoning about specific requirements for the adapted input method, the designer turned to the claims library to consider the intended psychological effects and specific implementation settings, as well as to obtain benchmark usability criteria for his own user testing. The claims library contained claims about several different input methods, including the ones his colleague designed and compared his design against, as well as several others that were part of the collection.

So far, the designer's activities with the design knowledge repository leverage basic "science of design" benefits present through a repository infrastructure.

Reflecting on the IRC ratings for each of these claims, he saw that no input method was able to achieve high reaction without also affecting either moderately high interruption or comprehension. In turn, the designer realized that the vehicle navigation system would be marketed to a broader demographic that would include users less willing to learn any complicated skill, suggesting that the graffiti-like input method would be inappropriate (it would cause unacceptably high interruption or reaction). The designer recognized an opportunity for developing an input method that could be used almost automatically, requiring very little attention or working memory access.

At this point in the case, the designer has recognized an opportunity for innovation, rather than adaptation, encouragement that the claims library may be a step in the right direction toward key focus point #1. The designer's realization was spurred by consideration of the three-dimensional design space suggested by IRC factors. He was able to identify that the required IRC factors (forming the design model) were different those targeted or achieved by existing methods. Although this realization could have emerged from user surveys, marketing analysis, or even user testing, these types of processes would have been more costly than the analysis performed by the designer—validating the concept and our implementation of key focus point #2.

Seeking inspiration for a new design to support automaticity (indicated by low interruption and high reaction goals) in the generic task of selection, the designer continued to browse the claims library. Instead of just looking for knowledge related to design of input methods, this time he focused on abstract search indices that indicated the generic design task and desired IRC values. He conveyed confidence in this search strategy.

Use of the abstract search indices provided some hope that key focus point #3 was being adequately supported. However, we note here that this designer was very experienced with Sutcliffe's generic task ontology.

Although several claims were returned by this search, the designer was disappointed that more claims were not available and that claim quality and evidence was inconsistent. Some claims summarized observations made in usability studies or presented in published papers, while the source of other claims was unclear.

Here, we start to see some tensions resulting from barriers to knowledge management (key focus point #5). In the version of the claims library that this designer was using, the process of adding claims to the system was an action designers performed (or, more often, did *not* perform) during the documentation phase of their project. Although administrators sporadically enforced claim quality, no mechanisms existed to encourage quality or submission.

However, one of the claims returned sparked an idea for the designer. The claim made reference to providing selection capability "at the fingertips of a user." When the designer extended the analogy to an in-vehicle navigation system, he thought of steering-wheel mounted controls that would allow a user to type alpha-numeric characters with two 8-directional pads manipulable with a thumb (thus, Thumb-Type). Character mapping decisions were made to promote ease of learning through the most intuitive orientation—it was assumed that users would be able to select characters as easily when they were driving as they could when they were focusing on the selection task. The designer developed a simple prototype in a few days, adequate for conducting testing of users driving in a simulator.

Based on the IRC benchmarks obtained with the original graffiti-like input method for the music selection task, user testing of Thumb-Type was discouraging. Actual user performance data indicated this particular implementation was not close enough to the design goals (supporting low interruption, high reaction, and low comprehension) to suggest continued development might lead to a valuable innovation. However, results and design rationale were archived as a point of comparison for alternate implementations, and the idea was brought to closure in a few weeks, rather than months.

Unfortunately, the design process outcome in this case did not lead to a successful design product, but in the "science of design" context, it was a valuable process. A relatively inexperienced designer recognized an opportunity for innovation and the initial design was ruled out expediently. Furthermore, the knowledge resulting from the initial design effort (otherwise unworthy of publication due to its failure), once archived in the claims library, would prevent another designer from making the same incorrect design choices and assumptions. Most importantly, this case exposed strengths and weaknesses of the claims library for supporting the activities noted in a few of the key focus points. The next case study looks at a design facilitated by an improved version of the design knowledge repository.

4.2. Case 2: NewsBar Notification

As preliminary work for an interface development bid, seven design teams (which consisted of 4-5 members, including industrial systems engineers, programmers, and HCI specialists) developed rapid prototypes of a notification system that could deliver news-related information to desktop computer users. The interface was envisioned to be part of a subscription service for premium news feeds—a persistent desktop interface client would ensure that the subscriber (user) stayed aware of late-breaking information that was essential to him, and could readily access full versions of the news content. Another essential design requirement was ensuring that the system would not be annoying to a user during short or long-term use (through any unwanted distraction or interruption), since that could impact satisfaction with their subscription.

Note that these user goals relate to desired (rough) levels of interruption, reaction, and comprehension, or IRC.

The seven development teams each pursued separate design proposals. They gathered detailed requirements through interview and focus group sessions with potential users, reflected on psychological tradeoffs for various design options, and developed limited-functionality prototypes (all seven designs displayed content from the same static source for a two-hour period of time—we were interested in comparing the effectiveness of the display techniques). A single testing team obtained performance metrics and subjective feedback for each prototype after it was used for several minutes by users engaged in other work tasks. Each prototype was also analyzed by three experts to determine its effectiveness at supporting the user goals.

Strengths and weakness of each prototype, as related to specific dominant design features, were identified, recorded as claims, and placed in the claims library. Since multiple designs showed strong development potential, the decision was made to have a new developer design another option, attempting to reuse several of the strongest features from the initial prototypes (specific features to reuse were not specified).

Before continuing, we discuss some decisions made to enlarge the concept of the claims library, which affected the new developer's approach. After observing the designer from Case 1 and other design projects, we were encouraged by success with following the process in Figure 1. We began to envision a design knowledge repository that included broader design process support activities through overall project archival, including identification of design objections and outcomes. Our general thought was that designers would normally access reusable design knowledge in the course of a specific design—if we could store data about their specific objectives, the repository components found useful, and repository contributions, then we might overcome some of the knowledge management barriers noted in Case 1.

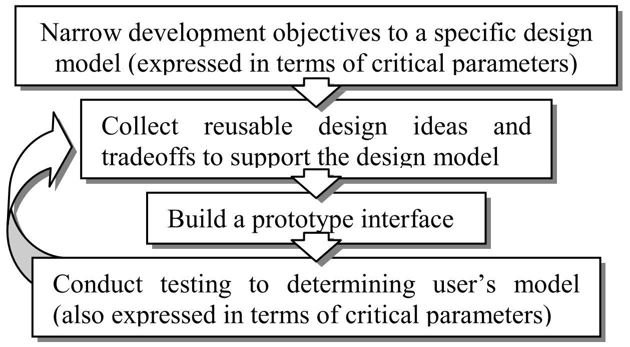

Narrow development objectives to a specific design model (expressed in terms of critical parameters)
Collect reusable design ideas and tradeoffs to support the design model
Build a prototype interface
Conduct testing to determining user's model (also expressed in terms of critical parameters)

Figure 1. Design processes surrounding use of a design knowledge repository, suggesting enlargement of the system to an *integrated design environment (IDE)*.

As we were exploring how to integrate project archival services with design knowledge repository access points, we asked the new developer (the continued subject of Case 2) to perform specific activities that would allow us to reflect on these services before they were fully implemented. Again, our motivation in observing the design process was to validate portions of our knowledge repository activity design, as well as uncover new breakdowns and areas for improvement.

First, the developer (an undergraduate programmer) proceeded through a few requirements analysis steps to narrow specification of the problem and facilitate its translation to abstract terms. After drafting scenarios to describe anticipated user interaction, the developer identified the generic tasks that users would perform and sequenced them as a hierarchical task analysis (HTA) within Norman's six stages of action (i.e., perceive, interpret, etc.)[13]. Next, he used a tool that had been developed to help designers obtain specific IRC ratings for their design model (described in [5]).

After using the IRC tool, designers are provided with a template that suggests how each critical parameter (i.e. interruption) could be expected to change as the user interaction transitions through the stages of action. These types of features (an extension of key focus point #2) were designed to address key focus point #4 and should be most useful to users that have difficulty with abstracting their requirements or finding generic design knowledge that might be applicable to their needs.

Once this process was complete, the developer was pleased that he was thinking about the design in very thorough and high-level terms. He admitted that going through the process made him abandon an idea that he initially had when first hearing about the design assignment (recognizing that it would not be suitable). At this point, he had a framework of generic tasks and IRC ratings to guide his search for suitable claims. He was able to identify many claims that influenced his design, to include several from the initial prototypes and a few others that had come from both analogous and very different systems.

This satisfactorily demonstrated how expanded design tools and services helped further key focus point #1 through clearer establishment of the design model. Coupling tools, services, and the design knowledge repository results in a web-based *integrated design environment (IDE)* prototyped as LINK-UP [5].

After the developer prototyped a new notification system design (NewsBar), an expert evaluator predicted how well the system would support user goals by estimating a user's model IRC rating, (using an assessment technique for the IRC critical parameters described in [4]). Comparing his design model with the user's model, the developer was able to instantly recognize that his design would not produce a high enough level of user reaction and trace the expected problem to specific claims and design features (which were included to produce appropriate reaction). He made several changes to both the design and his design model, noting that the combination of two claims, one pertaining to ticker rate and another related to relative size, could not be combined as easily as he first thought.

Through further effort (the entire effort took less than a week of full-time work), the developer was able to produce a design that experts determined would meet the user goals. While inspired by elements of other systems, it bore little resemblance to any in the end.

Here, we see validation for our focus point #1 strategy, as well as some ideas for resolving knowledge management tensions (focus point #5).

Reflecting on this case, the designer was able to craft a reliable notification system that would fulfill key user goals by synthesizing the experience of many other designers who had addressed similar problems. We expect that, through continued design work and design knowledge collection in a domain such as input methods for in-vehicle systems (the focus of Case 1), a designer would be able to follow the process illustrated in Case 2 for any domain to produce sound interface designs.

4.3. Case 3: Notifly

While there is certainly some value in enabling a design process that reliably produces *adequate* design products (fulfilling basic requirements for usability and utility), we aspire to a loftier goal—toward supporting *exceptional* interface design. Case 3 demonstrates such a design product and reviews the system support that played an integral part in its development.

A very successful design project emerged from a 15-week undergraduate seminar activity in which seven computer science students interested in HCI were challenged to develop notification systems. Although the system was required to deliver users information about airline flight prices that were available online, the instructors did not specify user goals that the system should support, but instead encouraged students to identify the critical parameter levels they thought were important, and to develop their systems accordingly. Students proceeded through the same requirements analysis steps identified in case 2: developing problem scenarios and generic hierarchical task analysis, as well as using the IRC design model rating tool.

Group discussions showed that students had a variety of different conceptions about the important goals. It was somewhat disappointing that they tended to cover all of the bases with their designs rather than identifying tradeoffs among the three critical parameters. Although the individual designs that the students developed did not gravitate to any distinct system class within the notification systems design space [10], promising features began to emerge in each.

Some of these features were identified through reusable claims in the claims library, while others were artifacts of the students' own invention. However, the most promising aspect of these features was that they embodied a strong tradeoff between the three parameters, even if the tradeoff was not supported by the system as a whole.

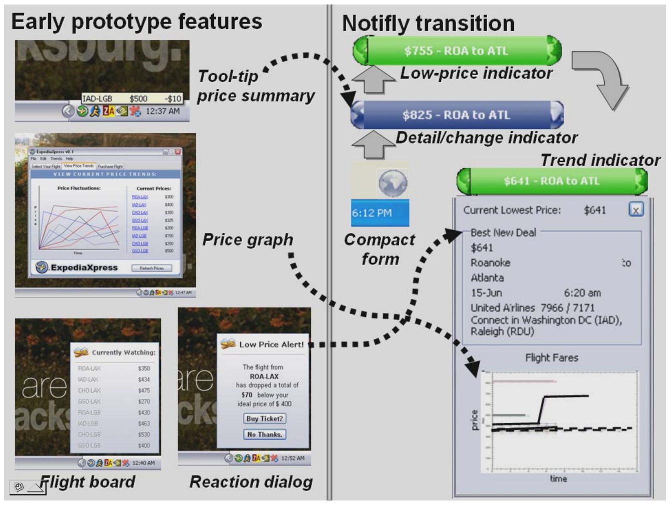
At this point in the case, the students were struggling to define strong design models (focus point #1), but the reasoning process based on critical parameters was starting to provide the necessary focus for design improvement (focus point #2). Recognizing this state, the instructor drew out discussion to highlight desired differences between the critical parameters students were targeting, often having students compare intentions. While some students were able to identify and reuse design artifacts successfully, they generally had more difficulty with the generic indices (key focus point #3) than designers in Case 1 and 2, but the recommendation features supporting key focus point #4 (described in Case 2 commentary) were not yet implemented within the IDE.

To help the students further realize some of the limitations in their designs, the instructor organized two usability evaluation processes, both using the integrated design environment (IDE). First, students transferred design rationale and screenshot depictions into the claims library through the IDE. Then, other students acted as anonymous expert reviewers and, with the help of a user's model IRC assessment tool [4], they provided each other with user's model ratings. Next, the designers each used a tool within the system to prepare and conduct an empirical test with a few participants, obtaining performance metrics in a dual task situation that allowed calculation of an actual user's model IRC. As expected, none of the designers were not particularly pleased with their results.

By conducting these evaluations through the IDE, not only were the designers facilitated in receiving feedback, but the design expectations and actual results became a permanent part of the claims library—a natural knowledge byproduct of the design process. This strategy effectively addressed key focus point #5, especially through the capture of poor design decisions (still an extremely valuable source of knowledge) that would not have normally been archived.

While the students reflected upon their individual design intentions and products, they were given the freedom to continue their development efforts as they saw fit—they could revise individual design models and interfaces or work in teams to improve a more promising prototype. An upcoming undergraduate research symposium provided motivation to produce the highest quality system possible, since the top submissions would receive large cash prizes. Therefore, it was somewhat surprising that, with only a month left before the symposium, all seven students decided to work together on a completely new system (although still a flight price notification system). They reasoned that they had the best chance of creating a high-quality system by reusing features from several of the prototypes to support four distinct design models that would correspond to user customization options.

Certainly, this was not the path of least resistance (especially working as a large team), nor a strategy that would position each individual with the chance for the largest cash award (any prize would have to be split in seven parts). However, the decision reflects the confidence students gained in the approach of selecting a distinct design model in terms of critical parameters and deliberately developing and reusing appropriate interface design components to match (key focus points #1 and 2).



IAD-LGB $500 -$10

$755 - ROA to ATL

$825 - ROA to ATL

$641 - ROA to ATL

Current Lowest Price: $641

Best New Deal
$641
Roanoke to
Atlanta
15-Jun 6:20 am
United Airlines 7966 / 7171
Connect in Washington DC (IAD), Raleigh (RDU)

Currently Watching:	
ROA-LAX	$350
IAD-LAX	$434
CHO-LAX	$475
GSO-LAX	$270
ROA-LGB	$430
IAD-LGB	$463
CHO-LGB	$530
GSO-LGB	$400

Low Price Alert!
The flight from ROA-LAX has dropped a total of $70 below your ideal price of $ 400
Buy Ticket?
No Thanks.

Figure 2. Screenshots of a Notifly transition and earlier prototype features that were reused in the redesign effort (indicated by dotted arrows).

As the team of students developed their new design, they drew from each other's claims, which were accessible through the claims library and supported with evaluation results (see Fig 2). They made rapid progress and generated a new prototype, which they validated with user testing just in time for the symposium.

We attribute much of the students' ability to rapidly organize their design goals and achieve consensus about interface decisions to the structure imposed by the design process. Continually questioning the design model IRCs when there was doubt about implementation options and referring to results obtained in earlier testing (apparent through the claims library) helped the students judge their own progress and readiness to move to new issues.

At the symposium, the resulting system, Notifly, was selected overwhelmingly by the approximately one hundred symposium attendees for the "People's Choice" award. In addition, the four groups of industry judges representing corporate program sponsors chose Notifly for the 2nd Place "Industry Choice" award.

Few at the symposium would have guessed that the award-winning Notify had been developed in such a short iteration cycle, or had been based on such different initial prototypes. Although there were many factors that may have influenced this outcome, we believe that the most dominant factor was the use of the claims library through the IDE processes. With this tool, the team was able to take advantage of reusable design knowledge and apply it in an innovative way that resonated with real people.

Table 1. Summary of our key focus points for design knowledge repositories, with connections to case observations.

Key Focus Points		**Case Observations**, *noted as interpretations of case events*			
	User Requirement to Support	*Exposes Initial Breakdown* →	*Demonstrates Effective Activities* →	*Affects Design Outcome* →	*Suggests Next Step for Design Tools*
1	Facilitate designers' specific motivation to reuse knowledge	Need to establish a clear and distinct design model *(Notifly)*	Systematic, tool-supported questioning of Norman's conceptual models *(NewsBar, Notifly)*	• Rapid rejection of design *(Thumb-Type)* • Guided successful dvlp effort *(NewsBar, Notifly)*	Implement process in Fig 1 as IDE services *(NewsBar, Notifly)*
2	Narrow the designer's focus of concern	*[established in prev work]*	Critical parameter analysis to focus on essential psychological effects *(all cases)*	• Prompted innovation *(Thumb-Type)* • Streamlined design decision making *(NewsBar, Notifly)*	Integrate critical parameters wherever possible into design tools *(all cases)*
3	Index components to allow broad application to many problems	High learning curve required to use for faceted search *(Notifly)*	Abstraction with Sutcliffe's generic tasks and domain specific critical parameters (i.e., IRC) *(all cases)*	Reuse happened thru abstract indices *(Thumb-Type,NewsBar)*	Rely on abstraction for system or expert constructed searches only *(all cases)*
4	Provide context to facilitate relation to generic structures	More support to traverse abstract structures needed for less experienced designers *(Notifly)*	Use design model and design activity context to infer finer-grained search indices *(NewsBar)*	Helped designer abandon incorrect initial assumptions *(NewsBar)*	Fully implement recommender features into IDE *(Notifly)*
5	Overcome barriers to knowledge management	Poorly developed knowledge content *(Thumb-Type)*	Archive reusable knowledge as a by-product of design activity *(Notifly)*	• Good design based on reusable content *(Thumb-Type,NewsBar)* • Structure facilitated groupwork *(Notifly)*	Tightly couple repositories with dvlp environments *(NewsBar, Notifly)*

6. Conclusions

Based on the case observations (summarized in Table 1), we are able to offer the following general guidelines, suggested by our successes and failures, for developing design knowledge repositories and their users' activities:

- **Consider the broad rationale-related requirements typical of knowledge management activities**, beyond lower-level activities analogous in software reuse,
- **Combine repositories with integrated design environments** to support foundational HCI processes and produce reusable knowledge with little extra effort,
- **Use domain-specific critical parameters** to guide continual design questioning of psychological effects,
- **Integrate system-driven recommendation features** to facilitate searches for abstract reusable components.

Of course, there are tradeoffs inherent in this approach. Since we have developed our repository and associated processes especially to enhance the work of novice designers, these guidelines may not be as applicable for seasoned designers or groups working within industry. The general design process built into our IDE constrains methodological approaches to design, but modules can be added to support other approaches. Our case studies must be followed up with more controlled research methods to compare outcomes with current approaches, as well as test cases that involve designers external to our lab. However, we are pleased with the initial progress we have made toward understanding how to develop research infrastructure to support a science of design. Since techniques and methods for teaching, engineering, and evaluating usability for interfaces like notification systems have not been fully developed and evaluated, we are eager to explore how a design knowledge repository like ours can be a catalyzing force.

References

[1] Christian F. Allgood. "The Claims Library Capability Maturity Model: Evaluating a Claims Library." Virginia Tech, (Master's Thesis), 2004.

[2] John M. Carroll and Mary Beth Rosson. "Getting around the task-artifact cycle: How to Make Claims and Design by Scenario." *ACM Transactions on Information Systems (TOIS) 10*(2), 181-212, April 1992.

[3] John M. Carroll, Mark K. Singley, and Mary Beth Rosson. "Integrating Theory Development with Design Evaluation." *Behavior and Information Technology 11*, 247-255, 1992.

[4] C. M. Chewar, D. Scott McCrickard, and Alistair G. Sutcliffe. "Unpacking Critical Parameters for Interface Design: Evaluating Notification Systems with the IRC Framework." In *Proc of the ACM Conf on Designing Interactive Systems (DIS '04), 10 pgs,* Aug 2004.

[5] C. M. Chewar, Edwin Bachetti, D. Scott McCrickard, and John Booker. "Automating a Design Reuse Facility with Critical Parameters: Lessons Learned in Developing the LINK-UP System." In *Proc. of the 2004 Intl Conf on Computer-Aided Design of User Interfaces (CADUI '04)*, 236-247, Jan 2004.

[6] Thomas Erickson. "Lingua Francas for Design: Sacred Places and Pattern Languages." In *Proc. of the ACM Conf. on DIS '00*, 357-368, 2000.

[7] Chuck Holbrook. "Input Methods for Notification Systems: A Design Analysis Technique with a Focus on Input for Dual-task Situations." Virginia Tech, (Master's Thesis), 2003.

[8] Charles W. Krueger. "Software Reuse." *ACM Computing Surveys 24* (2): 131-183, 1992.

[9] Ann Majchrzak, Lynne P. Cooper, and Olivia E. Neece. "Knowledge Reuse for Innovation." *Management Science 50* (2): 174-188, Feb 2004.

[10] D. Scott McCrickard, C. M. Chewar, Jacob P. Somervell, and Ali Ndiwalana. "A Model for Notification Systems Evaluation—Assessing User Goals for Multitasking Activity." *Transactions on Computer-Human Interaction (TOCHI) 10* (4): 312-338, Dec 2003.

[11] D. Scott McCrickard, Mary Czerwinski, and Lyn Bartram. "Introduction: Design and Evaluation of Notification System Interfaces." *International Journal of Human-Computer Studies 8*(5): 509-514, May 2003.

[12] William M. Newman. "Better or Just Different? On the Benefits of Designing Interactive Systems in Terms of Critical Parameters." In *Proc. of the ACM Conf. on DIS '97*, 239-245, 1997.

[13] Donald A. Norman. "Cognitive Engineering." In D. A. Norman and S. W. Draper, Eds., *User Centered Systems Design: New Perspectives on Human Computer Interaction*, 31-62, Lawrence Erlbaum Associates, 1986.

[14] C. Payne, C. F. Allgood, C.M. Chewar, C. Holbrook, and D. S. McCrickard. "Generalizing Interface Design Knowledge: Lessons Learned from Developing a Claims Library." In *Proc. of 2003 IEEE Intl Conf. on Information Reuse and Integration (IRI '03)*, 362-369, 2003.

[15] Ioana Rus and Mikael Lindvall. "Knowledge Management in Software Engineering." *IEEE Software 19* (3), 26-38, May/June 2002.

[16] Alistair G. Sutcliffe. *The Domain Theory: Patterns for Knowledge and Software Reuse*. Lawrence Erlbaum Associates, 2002.

[17] Alistair G. Sutcliffe and John M. Carroll. "Designing Claims for Reuse in Interactive Systems Design." *Intl Journal of Human-Computer Studies 50*, 213-241, 1999.