Ontology Models for Interaction Design: Case Study of Online Support

Keith A. Butler
University of Washington
Seattle, WA 98195
Keith.A.Butler@gmail.com

Anne Hunt
Primal Fusion
Toronto, Ontario, Canada
anne.hunt@primalfusion.com

John Muehleisen
Microsoft
Redmond, WA
johnmue@microsoft.com

Jiajie Zhang
University of Texas at Houston
School of Health Information
Science
Jiajie.Zhang@uth.tmc.edu

Beth Huffer
Sumaria Systems
Washington, D.C.
huffer.beth@gmail.com

Abstract

We report a case study for online self-support, which illustrates an advanced form of work modeling based on ontology technology. This new method enables a much earlier understanding of the design problem and promotes interdisciplinary design collaboration. A functional prototype was implemented for user testing and showed significant improvement in content discovery.

Keywords

Work-centered design, entity modeling, work ontology, information architecture, task analysis, faceted navigation and search, interactive problem solving, model-based design, methods, ontology modeling, representation effect

ACM Classification Keywords

H.5.2 [Information Systems]: Information Interfaces and presentation–User Interfaces- Theory & Methods

General Terms

Design, Human Factors, Theory

Introduction

We report the successful case study for interaction design in a work domain that is notoriously difficult for end-users: online self-support for malfunctioning personal computers (PCs). Our case study explores and illustrates a new method based an advanced form of work modeling. Summative user testing on a functional prototype of the design indicated that it is significantly more effective than the existing system ($p < 0.05$).

The new form of work modeling exploits ontology management tools (OMT) [20] to capture the fundamental characteristics of the work problem that the new design must satisfy. These characteristics provided stable, fundamental requirements from the analysis stage of the project through implementation of a fully functional prototype. The availability of the problem model enabled several strategically important improvements in design methods over conventional approaches: It allowed us to understand the design problem before making any major design decisions; It provided the conceptual model for the information architecture; It supported user interface (UI) design by serving as a reference model for the interactive representation of the problem; It enabled concurrent engineering with widespread collaboration among our interdisciplinary project team; The OMT model of the problem also provided reusable code for the application.

Early Understanding for the Design Problem
Understanding the problem that a new design must solve is fundamental for valid requirements. From a practical standpoint it is important for analysts to capture that understanding early in the project, in time to communicate it effectively before major design decisions are made that are expensive to change.

One popular technique for understanding the design problem is to model the work of users. Modeling and analyses of user work in one form or another is a fundamental step in virtually all the major design methods for HCI [e.g., 5; 15; 27] and there are dozens of forms of work modeling in the literature [8; 14; 18; 30]. By far, most are in the large family of *task analysis* techniques. In HCI task analysis is used to model the procedures that people will perform to accomplish work by interacting with computers or other complex devices. Task analysis can model the procedures of users working either singly or in teams. The procedures are represented in terms of action sequences, dependencies, and hierarchies. They may represent users' overt behavior, e.g., actions with input devices [8], or users' perceptual-cognitive processes [14, 17], or combinations. Task analysis informs design by making explicit the procedures that are needed to take advantage of the functions the system offers.

The strength of procedural models, however, also limits their use for early understanding of a design problem: A procedural model represents *the use* of functions. Kieras explained that procedural models embody fundamental design decisions about functionality and how it is allocated among users and devices [19]. Kieras also argued that two of the most fundamental questions that designers must answer are: What does the system do as a whole; and what role will the human operator play? The situation creates a paradox: If a design problem needs work modeling to be understood, and procedural models already embody important design assumptions, then a procedural work model can prematurely constrain the understanding in serious ways.

Methodologically, Vicente [32] argued it is more likely that a new design will realize better functional possibilities if it is not constrained by premature assumptions. Practically speaking, wrong choices about functionality are increasingly expensive to change after coding has begun.

However, by depending on procedural models exclusively we can inadvertently relegate important user input to a less important role in projects. Further, when project leaders make major design decisions without first understanding the problem that the new system must solve, they may end up deciding that the problem is whatever it is that their system does solve, and deliver something unusable.

Declarative Models of Work

The situation requires a new technique to model and analyze the problem that a design must solve in a way that is independent of the means to solve it. One important alternative to procedural modeling is *declarative* modeling. Cognitive theorists were among the first to recognize the importance of the distinction between knowledge that is conceptual versus knowledge that is procedural [1; 29]. Conceptual knowledge consists of declarative propositions, e.g., "Boston is a city in Massachusetts."

Patel & Kaufman [25] described how the distinction plays out in the design of HCI: Roughly speaking, declarative knowledge defines a specific application domain, while procedural knowledge represents how a user can perform a task with the application. Other authors have recognized the value of declarative models for software projects. Class diagrams are one form of declarative modeling that has been adopted for the analysis and design of software in the Unified Modeling Language (UML) [3]. They represent classes of objects and their relationships graphically for software engineers who are working on the analysis and design of object-oriented programs. More relevant to HCI design is the use of class diagrams to document an application domain in the first stage of the Use-Case Driven method for object-oriented software by Jacobson, et al. [16]. UML has a well known training example that shows how to analyze the application domain for a ticket reservation system by diagramming its objects and relationships, such as customers, shows, performances, tickets, etc. One limitation of class diagrams is that they can quickly become difficult to comprehend visually. It is not unusual to see large printouts of class diagrams covering dozens of square feet on walls in hallways where UML is being applied.

Modeling the Entity of Distributed Cognitive Work

We adopt the use the term "entity" here from industrial engineering [21]. Entity models are also declarative, but have a more focused scope and purpose than domain modeling in UML. When industrial engineers design work systems of people and machines to produce a physical entity, such as a wing bracket that is part of an airplane, that entity's specification is used as a fundamental requirement: If the new manufacturing system did not produce the specified part then it would be a failure, regardless of any other system qualities. We understand how a specification can be made in physical properties, such as shape, strength and weight, in a manner that is independent of the means to produce it. E.g., stamping, molding, or sculpting, could each create equivalent brackets. This important principle gives the system designer latitude to consider a variety of system solutions, and compare their qualities, such as cost, speed, or reliability.

Many important work entities in HCI, however, cannot be specified by physical properties. Instead they are cognitive entities, such as a decision, a classification, a diagnosis, or a plan. Until recently HCI has not had a technique to specify the entity of cognitive work in a manner that is independent of the means to produce it. To address this need we have adapted modeling techniques used in research on human problem solving in cognitive science. In particular, Zhang & Norman [33] used declarative ontology models of the entity of problem solving to prepare materials for their psychology experiments. They demonstrated how ontology models could be used to abstractly define the fundamental characteristics of a problem in a manner that is independent of any affordance, procedures, or even cognitive strategy.

We view declarative modeling of a cognitive entity as having great potential for an analogous technique to the way specifications are used by industrial engineers: We use it here to gain an understanding of the problem that a HCI design must solve early in a project, without building-in any assumptions about how the design must be implemented or used.

Ontology Modeling

Ontology modeling techniques use declarative propositions about a domain of conceptual knowledge. Once a rather obscure topic in philosophy, ontology has become important for cognitive science, the semantic web, and software engineering, especially in health informatics [e.g., 28].

Butler & Zhang, et al. explained how ontology modeling can be applied to HCI design [4], and they demonstrated how it can be used to analyze the work entity that an aircraft scheduling system must produce [6]. For the scheduling problem they used relational algebra to build an abstract model of a schedule, and then used it as a reference model to drive design for the computations and for the UI. Unlike class diagrams that try to capture an application domain, their model focused narrowly on the entity (work product) that a cognitive system for aircraft scheduling must produce. The model revealed the complex dependencies between scheduling an aircraft for maintenance and scheduling it for flying, and allowed them to understand how to integrate the two types of scheduling. With this specification they were able to decide how it could be produced via an effective distribution of processing between user and computer.

Butler & Zhang, et al. also argued that declarative work modeling does not replace or diminish the importance of task analysis, but can actually make it more powerful. They showed how the entity model could serve as a specification to evaluate the design's procedures. The evaluation focused on their ability to change the entity from its starting state to the required goal state.

Ontology Management Tools

In recent years several ontology management tools (OMT) have been implemented using knowledge representation techniques, such as first-order predicate calculus [e.g., 2 & 20]. Ontology models in OMT are made up of declarative propositions about objects and relationships, but OMT differ from class diagramming tools in several important ways. In OMT the visual representation is not the main manner of interacting with the model, it is mostly for team communication. Computational inference is one of the most important features of OMT. Once a declarative work model is defined in OMT it is a computational object that can respond to queries. OMT models can serve in the analysis of an application domain, and then be reused as a component of the application software. An additional benefit is that OMT inference capabilities can be used to build large parts of an ontology model by elaborating on existing propositions, thereby reducing significant manual labor and making ontology use more practical.

OMTs were originally intended to improve information retrieval for web-based search or to build the knowledge base of artificial intelligence applications [20]. However, we will describe a new use of OMT. In the following case study our use of OMT was to understand and model the *problem space* from a user's perspective before making any major design decisions. We then show how the OMT model was reused in several valuable ways: to promote interdisciplinary team collaboration; to generate the information architecture; to inform the design of the user interface; and as a computational component of the software in the new application.

Case Study of Online Self-support

Each year several hundreds of millions of users from dozens of countries visit http://support.microsoft.com to seek support and service. The content, and the experience of finding that content, were originally intended for computing professionals. However the majority of users now identify themselves as "home users" of personal computing (PC). The case study we report here was a response to the need to reinvent their experience when they come to the site for support to deal with a malfunction on their PCs.

The design for technical support should cover the entire user's experience, from the point where they first notice a malfunction through verification of successful procedures to deal with it. The focus of this case study is on the experience for discovering the content needed to deal with their problem. An integrated, companion project by Douglass-Olberg, et al. [9] improved the procedures of the content. Partial implementation can be viewed in the content now at the support site.

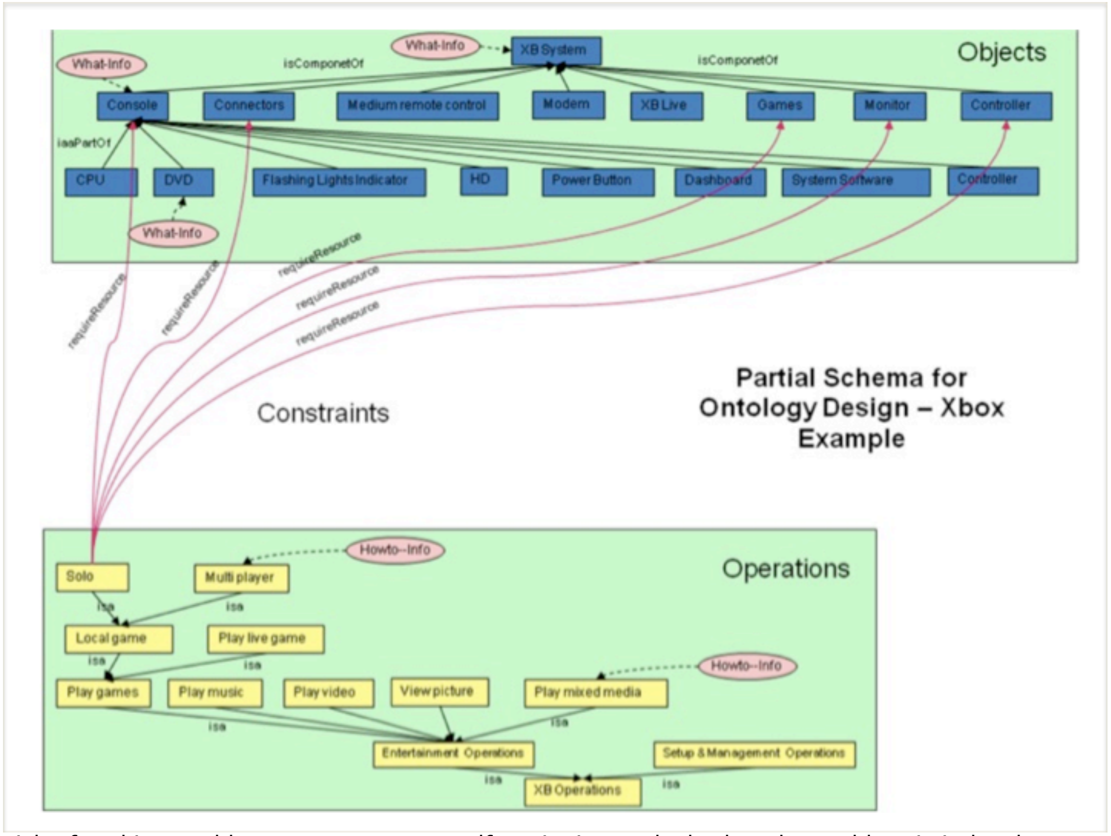
The following sections begin with a summary of contextual research findings, and then describe techniques and examples for modeling the user's problem space in OMT. We then explain how that model drove the selection of user interface technology and an integrated design effort that included the user interface, information architecture, and implementation factors. The section concludes with a summary of user testing results.

Background

We analyzed monthly data on the topics of calls to support centers and on user visits to support content on web sites. The data gave us problem trends and estimates of current success rates, but to help interpret them we need direct data on the current experience for home users.

We conducted half-day visits for contextual research [15] in the homes of about 50 home-users in seven major cities North America, Europe, and Asia. They provided us with work samples, inventories of software

Figure 1: Partial schema for OMT model



on their PCs, and common problem scenarios. We interviewed about problem solving strategies and resources, the values that motivate home users, and probed for technical knowledge about computing.

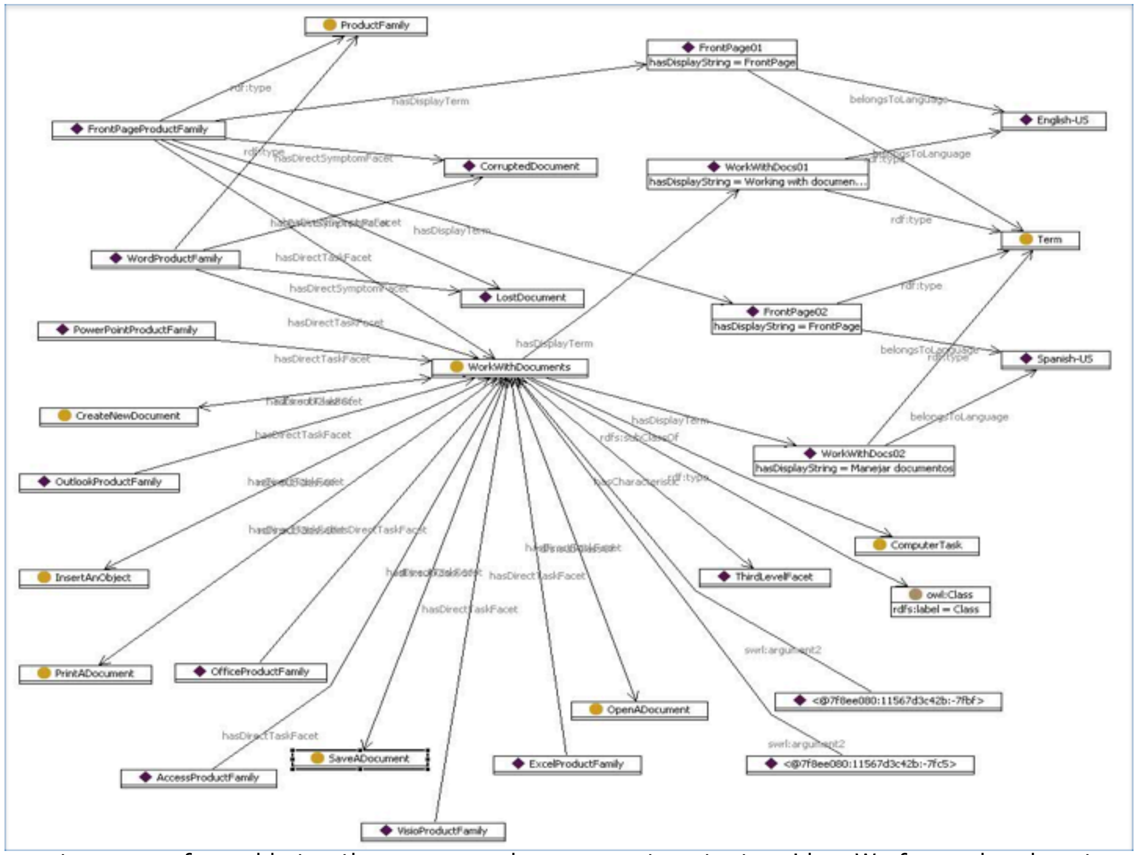
Digital photography is a very popular hobby for home users. They place great value their photos and other personal data but do not have convenient resources to back up data. Home users do have good problem-solving skills for online tasks like shopping. But they are hesitant about problem-solving for online self-support due to risk of making problems worse or even losing their valuable personal data. Home users often do not understand error messages and commonly try to close them or ignore them.

Home users do not typically have useful mental models of how their computers work. They have difficulty distinguishing which of the products on their PCs is malfunctioning and whether the problem is in hardware software. Consequently, they don't know which company they should turn to for support. They define their support experience as beginning when they notice a malfunction and concluding with verification of its repair. Home users can recognize the terms and feature names in applications and windows.

Home users typically have internet access, unless the malfunction disables browsers or networks. Those cases are a fairly small percent, but when home users seek support the most common resources are family, friends, or someone nearby who provides support services for a fee.

Our findings indicate that self-support needs to increase in the awareness and attractiveness of home users' decision-making. The online experience must compare favorably to other options in convenience and predictability, and by making a much stronger connection between the experience of a malfunction and the discovery of online content to deal with it.

Figure 0: Screenshot of portion of OMT model



Modeling the User's Problem

Our objective for this step was to understand the problem that home users encounter with the web space where support content resides. We focused on how to characterize the problem space and how this entity changes state as users progress towards a solution. We initially chose three representative products: Xbox, Word, and XP, with content sets that are small, mid-sized and large, respectively. This first group of products, and eventually all major home user products, were modeled using the TopBraid OMT (www.topbraidcomposer.com).

Figure 1 shows a partial schema of the model for Xbox, which is composed of *objects*, *operations*, and *constraints*. The operations are for tasks that consumers attempt with Xbox, and the objects (and their names) are part of the consumer's experience with the product. The knowledge for the user operations and objects was acquired by inspecting the UI of each product with experts. The model for each product provides, in effect, a high-level summary of product operations by users.

Unlike conventional ontology modeling, our model is based on the user tasks and the features of the products that home users could observe. In order to strengthen the association between product use and self-support we constructed the OMT model from aspects of the user's experience with the products: the task operations, visible objects in their experience, and terms that are recognizable to home users. Constraints and inference rules are important parts of the model, but they were provided from technical experts on the products, and not directly part of users' experience. The model also defined the space of suitable problems for consumers to attempt self-support. The models for all consumer products define the problem domain, and later were used to assemble a network of concepts that serves as the user's problem space.

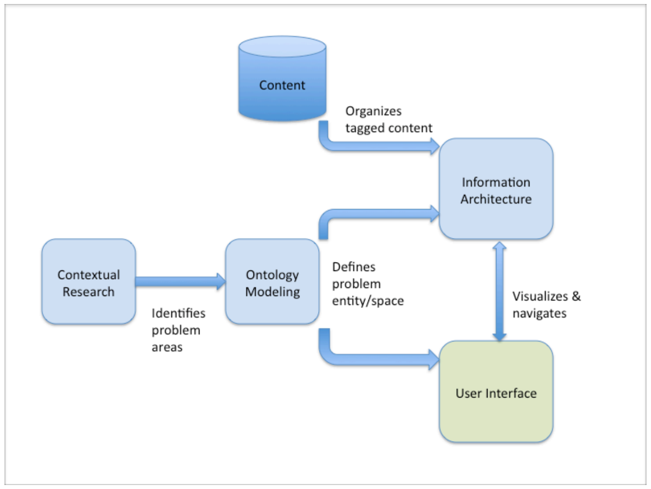
Figure 2 is a screen shot of part of the actual model for Xbox, as implemented in TopBraid. The model provides an underlying architecture for the information in the problem space. The model instantiates a graph, which is essentially a navigable network of pathways through the complex, interrelated objects of the product support domain. Support content was tagged to the terms of the OMT model for each product by a combination of automated and manual techniques. This was an important innovation to connect the user's problem solving, the problem space, and support content, to the user's experience with the product. We treated the problem space as an entity that users work on to arrive at the goal state- an intersection with the content they need. However, it is important to note that OMT is not intended for end-users. Home users interact with a UI that represents problem space, as described later.

Concurrent Engineering

A superior user-centered design can only impact users when it is complemented by effective technology-centered designs for implementation with network communication, data management, etc. Large, complex systems require the effective integration of design efforts from several disciplines.

Our first two steps gave us an understanding of the context and nature of the user's problem before we made major design decisions about how users should discover content. As shown in figure 3, the availability of the OMT model allowed us to make major design decisions in an unconventional sequence that promoted interdisciplinary collaboration. The OMT provided a common model for multidisciplinary collaboration across the teams for user interface design, information architecture, and technical infrastructure design. The work in these disciplines progressed in parallel using a concurrent engineering method [26]. The technology for the user interface and the infrastructure for implementing it were determined at an architectural level in parallel.

Figure 3: Unconventional project flow



In the following sections we explain how we chose the UI technology for an interactive visualization of the problem space, how we derived an information architecture from the OMT model to support the user's interactions, and how we integrated design to the detailed level by iterating among the disciplines.

UI Technology Selection

The then-current system constrained the user's task as a web search, however we view search as a technology, not as a type of user problem. In our approach the role of the UI is to provide an interactive representation of the problem space, to provide affordances to change the state of the space, and to provide visibility of the changes. Rather than the accepting the existing constraints, we first understood the problem space and then surveyed visualization technology for a suitable UI development tool that would allow the user to view and interact with the problem space that was defined in the OMT model.

We asked our designers and analysts to think in a way that was constrained only by the fact that the content is online, and by reasonable parameters for time and budget. We applied Zhang's technique for *representational analysis* [34] to the candidate technologies to narrow the list to those that were suitable to provide an interactive representation of the problem space. We then made our selection for one that could be implemented by a fairly inexpensive extension of the existing technical infrastructure.

We chose the Flamenco capability for faceted navigation and search for our users to view and interact with the problem space [12; 13]. Facets in Flamenco work well with a browser and they enable users to interact with a problem space by selecting combinations of the facets, which represent the dimensions that organize the space.

Information Architecture

Our information architecture provides a virtual organization to help users discover the content they need. An important factor is that support content is driven largely by accidental software failures in software products. The randomness of the failures makes it difficult to find organizing patterns within the large library of content that are understandable to home users. For example, if some form of cluster analysis were used, the contrasting dimensions might be interesting to a quality engineer, but are likely to confuse a home user. Our information architecture was derived from the problem space, and provided a structure for organizing content around dimensions that are understandable to home users.

We developed an information architecture that corresponds to the problem space by querying the OMT model. The queries were carried out using the native query capability in the TopBraid tool suite. The result of the queries formed a facet graph that governs the behavior of the facets. The graph required very little manual editing.

Facets provide users with an understandable path through their problem space. Given some user input, usually a mouse click on a facet the system responds with options that are neighbors in the facet graph. Facets are not strictly within the same taxonomic hierarchy. The facet graph is composed of thousands of facets and relationships among them. The definition and maintenance of these relationships was accomplished by modeling, in the OMT, via general business rules that were used to infer the facet-relatedness graph. We used an ontology-language representation of the associated business rules to produce a view of the data for use in faceted navigation and search. Examples of these rules include:
"Users interested in a product are interested in activities that can be done using that product."

"Users interested in an activity such as printing documents are interested in the hardware needed to do that activity."

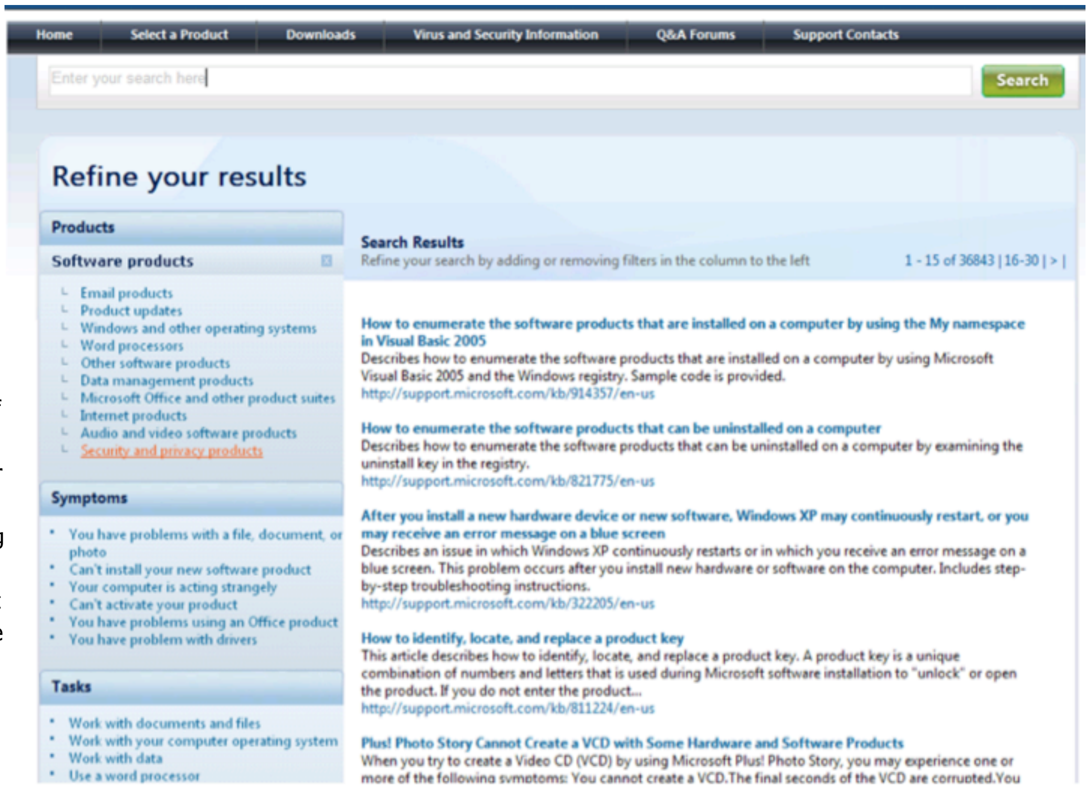
These types of rules can be adjusted as user requirements change, without affecting the basic model of products, user tasks, and product symptoms.

Our information architecture is virtual: It does not involve any changes to content. Instead, the support content was associated to the OMT dimensions through a combination of manual and automated tagging. Tagging consists of identifying a relationship between the terms of the OMT model and support content, and storing the results in a database. When the user selects a combination of facets the graph surfaces a set of tagged support content as results, where each item of content in the results set is one that was previously tagged, with the selected nodes. (Any node in the graph can be a tag associated with some content.)

Interaction Design
The management of both page navigation and results refinement is handled by the facet graph. The facets may appear as links or icons to navigate pages or as terms to refine results during search. In this technique the same facet graph integrates all user interactions with the problem space.

Facets provide the user with the ability to control a sequence of intersecting sets. In the example of figure 4, users first came to the home page and selected an icon, e.g., for security. The UI then navigated to a page for those products with relevant search results and more facets to refine the results.

As shown in figure 4, the result page consisted of three primary components: (1) the facets panel on the left, labeled as "Refine your results"; (2) The Search Results; (3) the Search box in which users can type additional search queries if desired. The facet panel provides users with an affordance to navigate and interact with the complex problem space by referring to their experience with the product. The facets' terms correspond to the user's experience with product use.



Refine your results

Products

Software products

- Email products
- Product updates
- Windows and other operating systems
- Word processors
- Other software products
- Data management products
- Microsoft Office and other product suites
- Internet products
- Audio and video software products
- Security and privacy products

Symptoms

- You have problems with a file, document, or photo
- Can't install your new software product
- Your computer is acting strangely
- Can't activate your product
- You have problems using an Office product
- You have problem with drivers

Tasks

- Work with documents and files
- Work with your computer operating system
- Work with data
- Use a word processor

Search Results

Refine your search by adding or removing filters in the column to the left

1 - 15 of 36843 | 16-30 | > |

How to enumerate the software products that are installed on a computer by using the My namespace in Visual Basic 2005

Describes how to enumerate the software products that are installed on a computer by using Microsoft Visual Basic 2005 and the Windows registry. Sample code is provided.

http://support.microsoft.com/kb/914357/en-us

How to enumerate the software products that can be uninstalled on a computer

Describes how to enumerate the software products that can be uninstalled on a computer by examining the uninstall key in the registry.

http://support.microsoft.com/kb/821775/en-us

After you install a new hardware device or new software, Windows XP may continuously restart, or you may receive an error message on a blue screen

Describes an issue in which Windows XP continuously restarts or in which you receive an error message on a blue screen. This problem occurs after you install new hardware or software on the computer. Includes step-by-step troubleshooting instructions.

http://support.microsoft.com/kb/322205/en-us

How to identify, locate, and replace a product key

This article describes how to identify, locate, and replace a product key. A product key is a unique combination of numbers and letters that is used during Microsoft software installation to "unlock" or open the product. If you do not enter the product...

http://support.microsoft.com/kb/811224/en-us

Plus! Photo Story Cannot Create a VCD with Some Hardware and Software Products

When you try to create a Video CD (VCD) by using Microsoft Plus! Photo Story, you may experience one or more of the following symptoms: You cannot create a VCD.The final seconds of the VCD are corrupted.You

Figure 4: facets as they appear in the results UI

Users can control facets to create a sequence of intersecting sets in the problem space by simply clicking on chosen facets. With each selection, the user can either choose content from the list of results, or make another selection from the updated set of browsing options. If the user makes another selection from the updated set of browsing options, the number of assets in the results set is reduced to only those assets that are tagged with *all* of the previously selected nodes *and* the newly selected node. With each additional selection, the results set is narrowed further until the user is left with a very short list of highly relevant documents, thereby obviating the need to sort through a large volume of irrelevant information, and increasing the likelihood that the needed content is among the results presented.

The facets' behavior is driven by the information architecture to give users an interactive representation of the problem space. Users see only their current portion of the problem space. This is a characteristic of a large set of facets, which in any configuration display only a portion of the problem space. Even with substantial interaction, users may not see the complete space, but our test data indicate that it is not necessary for users to see the entire space in order to discover content effectively.

Prototype for Design Validation

The purpose of the design validation was twofold: (1) determine empirically whether users perform their procedures as intended; and (2) determine if the new design increases the success rate for content discovery over the then-current system (baseline).

Facets allow users to select any permutation of terms, making it too difficult to conduct an evaluation with any type of simulated user interface. To support design validation we developed a functional prototype of software with about twenty-thousand pieces of tagged content using Flamenco [12; 13]. The validation study was a between-subjects experiment, where the independent variable was the version of the system: the prototype vs. the then current baseline version of the support web site. Each test condition collected data from twenty-two participants, who attempted same set of benchmark scenarios in randomized order. Each group had 22 participants who completed the testing. They were recruited from the consumer segment of a regional usability test services' participant database. The participants were classified primarily as home computer users. Participants were rewarded with free software upon completing the study, regardless of their results.

The problem scenarios were chosen from the 100 most-frequent reported problems to call centers, across a variety of support-task types. Each participant's computer was deliberately "broken" before they arrived. They were instructed to perform a task that would encounter the planted malfunction, e.g., "You want to download pictures from the web to make a Valentine's Day card." When they could not perform the initial task, they were asked to solve the problem of how to fix the computer by going to one of the two web sites for their experimental condition. The goal state was satisfied if they discovered and recognized the content to fix the problem. They were not required to actually complete the repair.

Summary of Test Results

The most important result from the prototype test was that users enjoyed a 33% increase in solution discovery over the baseline version ($p < 0.05$). In addition, the prototype had 52% fewer "give-up" trials, 74% fewer search queries, and non-significant trends for fewer false-positives and fewer overall user actions. Another encouraging outcome was the degree to which study participants attended to and interacted with the facets. The UI induced virtually all users to follow the user procedures as intended, whether or not they discovered the solution. The only recorded measure where the prototype was not better was its increase for false negatives ($p < 0.01$). Recorded comments indicate this is probably due to users attending to the facet panel after the target had already appeared in the results.

Conclusions & Discussion

It is important to evaluate new methods for analysis and design with objective data. We offer here a second case study, this one with a measurably better design solution, as shown by head-to-head testing against the then-current system. We believe the measurable improvement resulted largely from understanding the problem independently of the means to solve it before making major design decision. We recognize many other factors could play a role in these positive results, such as facets or a highly skilled team. However, the project unfolded largely as planned and the OMT model played a pivotal role.

Conclusions for the Application

The main conclusion for the application design is that technical self-support by consumers can be significantly improved by providing them with an interactive visualization of their problem space. Earlier designs for online self-support were based on a more narrow view that the users' task was essentially to search and find relevant information about their problems. In our design the users' task is defined more broadly as problem solving, which begins when they notice a possible malfunction. We intended facets to provide a representation of the problem space that allows users to recognize how their experience with a product is related to discovering the support content they need.

Related Work

Some forms of cognitive modeling in HCI also use conceptual knowledge. Cognitive modeling has been proposed as a technique for understanding the complexity of the user's device [26]. Our use is different and follows the distinction proposed by Long & Dowell. They view interactive applications as joint human-machine work-systems that are constrained by the application domain, but separate from it [10; 22]. Following this framework Dowell developed a formulation of the cognitive design problem for air traffic management [11] based on a three-dimensional model of aircraft separation. In some important sense the entity of air traffic management is the preservation of required aircraft separations in three dimensions. In this manner the entity can be defined independently from any particular machine or system design to perform the cognitive work of air traffic management.

Our use of OMT extends their concepts in several ways: To make a clearer connection to theory in cognitive science; to make the practice of design more efficient; and to move towards integrating HCI design with the other disciplines that it depends on for implementation. This distinction also complements conventional work modeling practice for HCI design, which typically focuses on procedural models.

Descriptive vs. Prescriptive Models

Conceptual modeling can be used in two ways: to describe how people think, or prescribe a better way for them to think. This distinction is also observed in research on decision-making, where prescriptive decision processes are designed to compensate for biases that often interfere with good practice [31]. Our use of conceptual knowledge modeling is prescriptive for two reasons. The model's purpose is to induce more efficient problem solving in users than their current understanding allows. Also, home users could not be expected to know about the entire problem space in our model because it covers so much detail on so many products.

Distributed Cognition and External Representations

We view interactive computing is an important type of distributed cognition [33; 34; 35]. Ontology modeling was used in research on distributed cognition to prepare materials for problem solving experiments. By deriving each representation from the ontology they assured that each was isomorphic (logically equivalent) to the problem ontology.

Our use of entity modeling is analogous. We treat the user interface as a form of external representation of the user's problem. Our use of ontology is to assure that the user interface and supporting functionality provide an accurate, interactive representation of the problem. Its value proposition is to re-distribute cognitive processing to be more effective [7], while not penalizing the user with inefficient tasks.

More Economical Designing

Our approach is strongly influenced by well-established methods for industrial engineering where complex human-machine systems are designed to produce physical products. A specification of the physical product that the new system must produce routinely drives design in industrial engineering. In an analogous way we used OMT to specify key characteristics of the entity (product) of distributed cognitive work. We call this approach *work-centered design*.

Without a clear specification of the entity that a system is supposed to produce the designer is pressured to over-supply functions and features in an attempt to make certain the needed entity is among the things it does produce. We see this situation in the current state of the art for interactive software engineering. Many software products include features that are rarely used but expensive to encode. After the product is released the inefficiency takes a new form: un-needed functions and features impose a severe form of uncontrolled cognitive overhead on users that interferes with productive work. Entity models can capture important, stable requirements that are independent of the context in which the work is performed or the technology that assists it [4]. We believe OMT can help designers make better decisions about the need for features and functions.

DESIGNING FOR NEW TYPES OF WORK

The inefficiency of design is likely to get worse. Early in the evolution of business computing there were many valuable applications that automated portions of human work that were well understood but too time consuming, error-prone, etc. For these applications designers had the benefit of familiarity to help them understand the nature of the deign problem. More recently the value of computing has moved towards applications that create new forms of work that could not have existed without computing. Without some way to understand the problem that a design is supposed to solve the risk of project failure will likely increase. We believe declarative modeling of the cognitive work entity will become more important as computing invents more new, important kinds of work.

COMPLEMENT TO PROCEDURAL WORK MODELING

Declarative work models do not replace procedural models. They have a different purpose. One way they complement is that a declarative model of the entity of cognitive work should serve as the primary evaluation criterion for procedural models. Different design procedures could be compared in terms of their qualities, such as usability or cost. But in order to be comparable the procedures must produce the same work entity. We are currently planning research on how performance models [e.g., 17] can provide measures of efficiency to compare verified, alternate designs.

Applicability

Our approach was developed for interactive, technical problem solving. Its applicability to other types of cognitive work has yet to be studied. We recommend it for applications where a clear definition of success can be defined and the problem is technical, large, complex, and valuable. There is legitimate concern about the cost of such a thorough analysis, but it important to note that the constraints of the entity on the success of a design will be in effect whether or not a project makes them explicit. Consequently, it can be far less expensive than the hit-or-miss tactics of many projects. It should also be valuable to managers who lead software projects for high-stakes systems, such as health-critical or safety-critical applications.

ACKNOWLEDGMENTS

We thank George Robertson, Susan Dumais, Marty Hearst, and Axel Roesler, for their valuable technical advice and encouragement. Research on the role of ontology in design was supported by USAF/AFMC Air Force Research Lab contract F33615-03-2-6315, Work-Centered Support Systems for Autonomic Logistics, 2004-2007. POC: Capt. Mike Blake, telephone (937) 255-8049. The case study material was released for publication by Microsoft on October 6, 2008. POC: D. Shannon, dshannon@microsoft.com.

References

[1] Anderson, J.R. The architecture of cognition. Cambridge, MA: Harvard University Press, 1983.

[2] Angele, J. & Sure, Y. (Ed.s) Evaluation of Ontology-based Tools. Proceedings of the OntoWeb-SIG3 Workshop at the 13th International Conference on Knowledge Engineering and Knowledge Management EKAW 2002. Published as CEUR-WS Vol-62 http://ceur-ws.org

[3] Booch, G.; Rumbaugh, J.; & Jacobson, I. The Complete UML Training Course. Prentice-Hall, 2000.

[4] Butler, K. A. and Zhang, J. (2009): Design models for interactive problem-solving: context & ontology, representation & routines. *Ext. Abstracts CHI 2009*, ACM Press (2009), 1-2. 4315-4320.

[5] Butler, K.A. Usability Engineering Turns 10. *Interactions*, 1, (1996), 59-75.

[6] Butler, K.A.; Zhang, J; Esposito, C; Bahrami, A.; Hebron, R.; & Kieras, D. Work-centered design: a case study of a mixed-initiative scheduler. In: *Proc. CHI 2007*, 747-756, ACM Press, 2007.

[7] Card, S.K.; Piroli, P.; & Mackinlay, J. The Cost-of-Knowledge Characteristic Function: Display Evaluation for Direct-Walk Dynamic Information Visualizations. In: Card, Mackinlay, & Schneiderman (Ed.s) *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann. 582-588, 1999.

[8] Diaper, D. & Stanton, N. (ed.s). *The Handbook of Task Analysis for Human-Computer Interaction*. Mahwah, NJ: Erlbaum, 2004.

[9] Douglass-Olberg, C., Farkas, D., Steehouder, M., Karreman, J., Kieras, D., Resoler, A., Dalal, N., Baker, R. & Brunet, D. The new face of procedural content: a real world approach. *Ext. Abstracts CHI 2008*, pp. 3495-3500, ACM Press, 2008.

[10] Dowell, J., & Long, J. Towards a conception for an engineering discipline of human factors. *Ergonomics*, 32, 1513-1535, 1989.

[11] Dowell, J. Formulating the Cognitive Design Problem of Air Traffic Management. *Int. Journal of Human- Computer Studies*. 49(5), pp. 743-766 1998.

[12] Elliot, A. Flamenco Image Browser: Using Metadata to Improve Image Search During Architectural Design. In *Proc. CHI 2001*, ACM Press, 2000.

[13] Hearst, M. Next Generation Web Search: Setting Our Sites. *IEEE Data Engineering Bulletin, Special issue on Next Generation Web Search*, Sept., 2000.

[14] Hollnagel, E. *Handbook of Cognitive Task Design*. Mahwah, NJ: Erlbaum, 2003.

[15] Holtzblatt, K. & Beyer, H. *Contextual Design.* Morgan-Kaufmann, 1998.

[16] Jacobson, I., Christerson, M., Jonsson, P., & Overgaard, G. *Object-oriented software engineering - a use case driven approach*. Reading, MA: Addison-Wesley 1992.

[17] John, B. E., & Kieras, D. E. The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction*, 3, 320-351, 1996.

[18] Kirwan, B. & Ainsworth, L.K. *A Guide to Task Analysis*. London: Taylor & Francis, 1992.

[19] Kieras, D. E. Task analysis and the design of functionality. In A. Tucker (Ed.) *The Computer Science and Engineering Handbook* (2nd Ed). Boca Raton, CRC Press, 2004.

[20] Knublauch, H. An AI Tool for the Real World: Knowledge Modeling with Protégé. JavaWorld.com, 06/20/03.

[21] Law, A. and Kelton, W.D. *Simulation Modeling and Analysis, Third Edition*, McGraw-Hill, 2000.

[22] Long, J. & Dowell, J. Conceptions of the Discipline of HCI: Craft. Applied Science, and Engineering. In: *Proc. Fifth Conference of the BCS HCI SIG*. Cambridge University Press, 1989.

[23] Newell, A. and H.A. Simon. *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall, 1972.

[24] Parsaei, H. R. & Sullivan, W. G. *Concurrent engineering: contemporary issues and modern design tools.* London: Chapman-Hall, 1993.

[25] Patel, V. & Kaufman, D. Cognitive Science and Biomedical Informatics. In: E. Shortliffe & J. Cimino (ed.s) *Biomedical Informatics, Third Edition*. Springer. Pp. 133-85. (2006)

[26] Payne, S.J. (2003) User's Mental Models: The Very Ideas. In: J. Carroll (Ed.) *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science*. Morgan-Kaufmann Pp. 135-156.

[27] Preece, J; Rogers, Y.; Sharp, H.; Benyon, D.; Holland, S.; & Carey, T. *Human-Computer Interaction*. Reading, MA: Addison-Wesley, 1994.

[28] Rosse, C. & Mejino, J. L. A Reference Ontology for Biomedical Informatics. *J. Biomedical Informatics*, 36(6), pp. 478-500, 2003.

[29] Rumelhart, D. E., & Norman, D. A. Representation in memory. In R. C. Atkinson, R. J. Herrnstein, G. Lindzey, & R. D. Luce (Eds.), *Stevens' Handbook of Experimental Psychology, 2nd edition*. New York: Wiley, 1998.

[30] Schraagen, J.; Chapman, S.; & Shalin, V. (ed.s), *Cognitive Task Analysis*. Mahwah, NJ: Erlbaum, 2000.

[31] Skinner, D. *Introduction to Decision Analysis*. Probabilistic Press, 1999.

[32] Vicente, K. Work Domain Analysis and Task Analysis: A Difference That Matters. In: J. Schraagen; S. Chapman; & V.Shalin (ed.s), *Cognitive Task Analysis*. Mahwah, NJ: Erlbaum, 2000.

[33] Zhang, J. & Norman, D. Representations in distributed cognitive tasks. *Cognitive Science*, 18, pp. 87-122, 1994.

[34] Zhang J. A representational analysis of relational information displays. *International Journal of Human-Computer Studies*, 45, 59-74, 1996.

[35] Zhang, J. External Representations in Complex Information Processing Tasks. In: A. Kent, (ed.), *Encyclopedia of Library and Information Science*. Marcel Dekker, New York, 2001.