Journal of Universal Computer Science, vol. 8, no. 6 (2002), 634-643
submitted: 4/2/02, accepted: 3/5/02, appeared: 28/6/02 © J.UCS

Usage-Centered Interface Design for Knowledge Management Software

Harald Karner
(Hyperwave R&D, Graz, Austria
hkarner@hyperwave.com)

Georg Droschl
(Hyperwave R&D, Graz, Austria
gdroschl@hyperwave.com)

Abstract: In IT-supported knowledge management (KM), the software user interface is at the boundary between persons and the knowledge management system (KMS). It plays a central role because seen from the users point of view, the user interface *is* the system. This paper presents a case study in which a particular User Interface Design methodology was employed to design a prototype KMS user interface for an inbound call center. In this example, we combine knowledge re-use and expert location.
Key Words: User Interface Design, Usage-centered Design, Knowledge Management, Interaction Patterns, Call Center
Category: H.5.2

1 Introduction

When it comes to KM, it is likely that software support will be required, especially when there are high degrees of complexity (having more information that you can manage or understand), and/or uncertainty (not having enough information) [Zack, 1999]. In practice, most KMS are *generic* in the sense that they are adapted to the requirements and needs of the special business case, rather than re-building software "from scratch" every time. [Hyperwave] is a popular KMS, and comes with a set of components, that system integrators may draw from when customizing the system. In this paper, we employ a usage based methodology to adapt the user interface to the needs of the individual case. For illustration, we show how to design the KMS user interface of a simplified inbound call center which enables us to demonstrate a combination of two distinct approaches to IT-based knowledge management: *person-to-person* and *people-to-documents*.

2 Usage-centered User Interface Design

[Constantine et al, 2000] introduced a model driven process to develop a realistic prototype of a software user interface. This process is based on three primary abstract models (role model, task model, and content model) and two supplementary models (domain model and the operational model) (c.f. [Figure 1]).

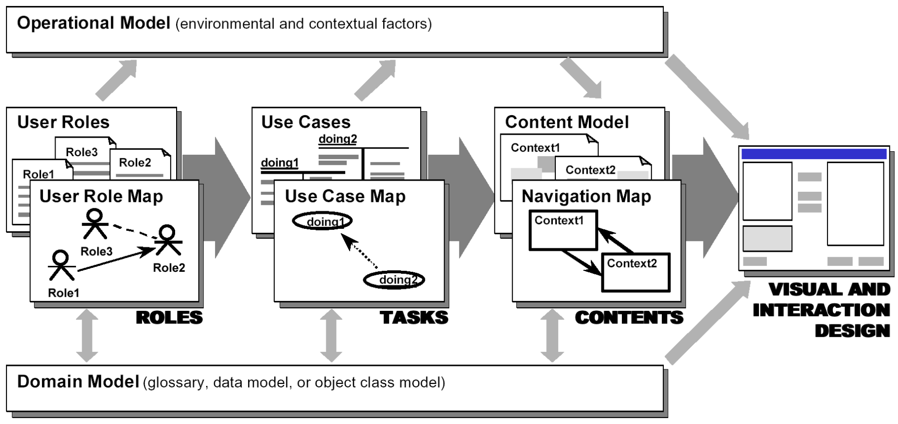


Figure 1: Logical relationships among primary models in usage-centered design

1. The *Role model* describes the relationships between particular users (roles) and the system being designed. This model consists of a set of user role descriptions and the user role map. Typically, a user role expresses the users' characteristic needs, interests, expectations, and behavior. A user role description may, for example contain information about the user's general computer skills, domain knowledge, system knowledge, frequency of use, etc. It should also contain the usability objectives for this particular role, like efficiency, reliability, learnability, user satisfaction, etc. A structured user role form, that allows to capture relevant aspects of an user role very rapidly can be obtained from [ForUse]. The user role map is a diagram that shows all user roles and the relationships between them. For example one role could be a specialization of another role, one role could be composed of other roles, and roles may depend on each other.
2. The *Task model* contains task descriptions in the form of so called 'essential use cases' as well as a use case map, that gives an overview of how these use cases relate to each other. An essential use case is a sequence of interaction between user and system, focusing on the purpose of interaction. It is written in the form of a simple narrative dialogue, using a two-column structure separating user intentions from system responsibilities. This structure has two advantages: First, it emphasizes the system boundary (i.e. the user interface), and second, it separates internal from external requirements. The use case map describes relations among the use cases.
3. The *Content model* or *Abstract Prototype* represents the content and organization of the user interface. It is modeled as a set of interface content models and a context navigation map. The context navigation map shows the possible transitions between the particular interaction contexts. An interaction context can be a page, screen, dialogue, or some other part of an user interface. The interface content models describe the content of the various contexts within which users interact with the system. A content model consists of abstract tools and abstract materials, that are arranged into particular interaction contexts. Abstract tools can

be commands, operations, operators, or other active components, that cause an action of some sort. Abstract materials can be data, information, containers, or objects that are operated upon or affected by tools.

4. Derived from the content model, the *visual and interaction design* now is a realistic prototype of how the user interface could look and behave. Here, the tools and materials of the content model are translated into concrete user interface widgets, interaction contexts may transform to pages, screens, dialogues, etc. The focus of visual and interaction design lies in selecting or designing and arranging the visual elements or user interface controls as well as in establishing the behavior of the user interface and how the users interact with it. This is also the point, where user interface design and usability guidelines should come into play.

During the construction of the visual and interaction design, usability guidelines should be applied to make sure that the resulting user interface matches the usability objectives stated in the role model. An alternative approach to using guidelines (‘interaction patterns’) is proposed by [van Welie, 2001]. Simply put, an interaction pattern is a reusable and proven solution to a problem in a specific context. The main parts of an interaction pattern are – apart from a meaningful and catchy name – a problem description, the context, and the solution. The context describes the characteristics of the users, tasks, and environment for which the pattern applies. It also provides criteria for deciding *when* this pattern shall be used (for example, for frequent users of the system). A collection of interaction patters can be found in [van Welie and Trœttberg, 2000].

The three core models do not have to be developed in a strictly sequential way as implied by [Figure 1]. As such, the set of models will aid the modeler’s understanding of the problem.

3 Example: Call Center

To illustrate this methodology, we will now show how the KMS user interface of a simplified inbound call center might be designed. In this example, calls are initiated by the customer (e.g. to request product information), as opposed to an outbound call center, where calls are initiated by the call center agent (e.g. to investigate customer satisfaction). The example enables us to demonstrate a combination of two distinct approaches to IT-based knowledge management [Hansen et al, 1999]:

- *Person to person* refers to developing networks for linking people so that tacit knowledge can be shared. It mainly applies to individual, and high-level problems by channeling individual expertise. KMS provide modern communications- and people location technologies.
- The emphasis of *people to documents* is on developing an electronic document system that codifies, stores, disseminates, and allows reuse of documented knowledge. It provides high-quality, reliable, and fast KMS implementation by reusing codified knowledge.

A call center can be seen as a combination of both: the caller requesting support for a specific problem may be provided with a document describing a previously solved, similar problem. In those cases where it is not possible to refer to documented problem-solving knowledge, the caller must be put in contact with an expert. Once successful, the solution is subject to documentation and future “knowledge re-use”.

Furthermore, call centers exist in many ‘vertical industries’, both inside organizations (for example, to respond to inquiries by employees), as well as to support customers, partners, and others.

3.1 Role Model

In our example, the role model contains the following roles:

1. *A Caller* making a call which will be answered by a call center agent.
2. The responsibility of the *Call Center Agent* is to answer incoming calls from callers. The caller then confronts the agent with a problem, who shall find a solution or a problem solver through the KMS. The Identification of an appropriate expert for the particular problem is also done via the KMS. Upon solving a new problem, it is the duty of the agent to document the case, so that the solution is available for reuse. Call center agents are expected to are knowledgeable of the KMS, since they are using it very often. This is why the main usability aspect for this role is efficiency.
3. From the point of view of the call center, the main aim of the *Expert* is to solve new problems. The expert makes use of the KMS to view background information about the caller, and the problem description. The expert has good computer knowledge, but only uses the call center KMS occasionally. Thus, the usability objective is ease of use rather than efficiency.
4. The *Web Interface User* is a variant of the Caller role, and makes use of a web interface rather than a telephone to have a problem solved. This leads to direct interaction with the KMS, without assistance of a call agent. The web interface user typically has basic computer and web usage skills, but has not necessarily interacted with this particular KMS before. Thus, the usability objectives for this role are ease of use, and comprehensibility.

The user role map [Figure 2] shows the relationships among the user roles. *Focal Roles* are those that interact with the KMS, and will be considered in subsequent models. For the sake of brevity, we will not model the expert. The Caller interacts with the Expert directly or through the Call Center Agent.

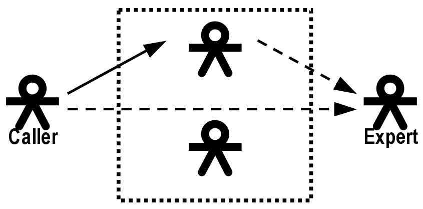


Figure 2: User Role Map

3.2 Task Model

For the role of the Call Center Agent, we identified the following three essential use cases:

1. *Serve a Caller*

User intention	System responsibility
identify caller	
	display caller information
[optionally] change caller information	
	Save changes
request problem description form	
	display problem description form
enter information about problem	
start search for previously solved problems	
	display search results
select single search result [optionally repeated]	
	display document describing solved problem

Table 1: Serve a caller essential use case

2. *Document solved problem*
   Precondition: Use case *serve a caller* carried out

User intention	System responsibility
	display problem description entered in *serve a caller* use case display search results of *serve a caller* use case display form for describing solution
read problem information read search results solve problem (assumed to be successful) enter description of how problem was solved	
	save the description of the solution along with relevant problem information and search results

Table 2: Document solved problem essential use case

3. *Transfer problem to an expert*
   Precondition: Use case *serve a caller* carried out

User intention	System responsibility
	display list of potential experts
select expert [optionally repeated]	
	display expert’s core competencies
transfer called to expert	

Table 3: Transfer problem to an expert essential use case

For the web interface user there is only one use case:

1. *Search for problem solution*

User intention	System responsibility
submit identification information (e.g. name)	
	display background information (e.g. products used by customer)
Specify potentially affected product(s)	
	display problem description form
describe problem	
start search for previously solved problems	
	display list of search results
select search result [optionally repeated]	
	display selected document

Table 4: Search for problem solution essential use case

The use case map [Figure 3] shows the relationships between the use cases [Table 1-4]. The use cases entitled ‘Transfer problem to an expert’ and ‘Document solved problem’ both depend on the ‘Serve a caller’ use case, because it has to be carried out before one of the others can take place.

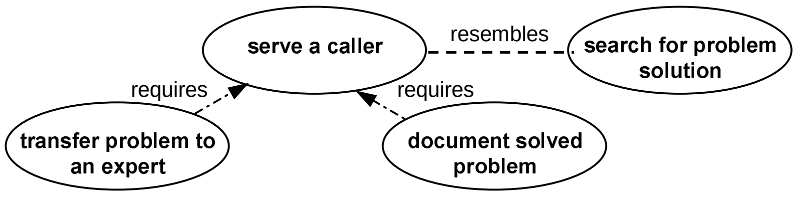


Figure 3: Use case map

3.3 Content Model

The interaction contexts in the content model will be displayed in a graphical notation entitled *canonical abstract layouts*. They are made up of *canonical abstract components* (in many cases a combination of boxes, arrows, and other basic symbols), that are a “toolkit” of abstract components for abstract prototyping. A *container* is graphically represented by a simple square, while an *operation/action* is visualized by an arrow. Special cases of the container element include *collection* and notification. Examples for operation/action are *start*, *quit*, *select*, *create*, *delete*, *modify*, etc. The advantage of the use of canonical abstract components is, that it allows the abstract layouts to be built more easily and naturally, and it as well narrows the gap between the abstract layout and the final visual design.

The ‘Serve a caller’ use case contained six intentions of the Call Center Agent which can be grouped into three interaction contexts. The first interaction context is to identify the caller, and to optionally change the background information. The second one to request and fill in a problem description form, and to search for previously solved problems. The final interaction context is to display search results. In terms of the graphical notation the caller information interaction context [Figure 4] is represented by a *container*, that for example holds another container entitled ‘General Information’, which contains four *elements* (Name, address, phone, e-mail), that can be modified.

1. *Caller information*

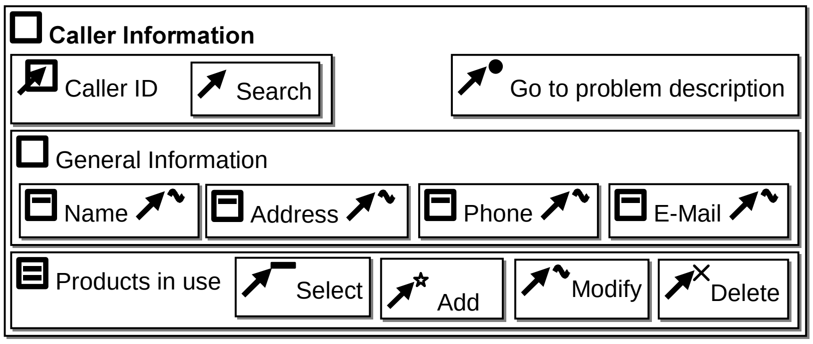


Figure 4: Canonical abstract layout for the caller information interaction context

2. *Problem description*

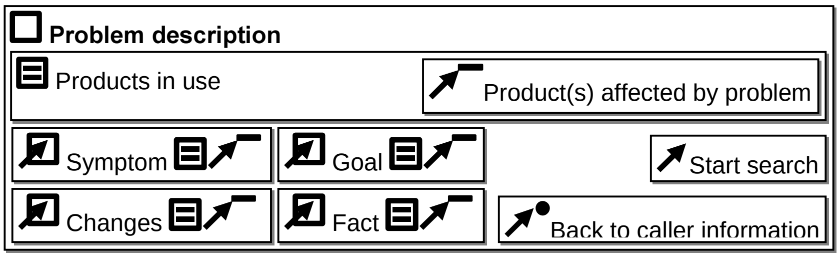


Figure 5: Canonical abstract layout for the problem description

3. Search results

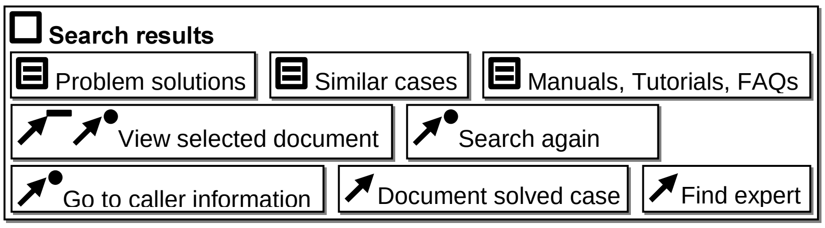


Figure 6: Canonical abstract layout for search results

The context navigation map [Figure 7] shows the possible transitions between the particular interaction contexts and by which action they are triggered. For example, by choosing 'Start search' in the problem description context will lead to a transition to the search results context. The map has been reduced to the interaction contexts, that are covered in this section. Each rectangle represents an interaction context, the arrows represent the actions, that lead to an context transition. A double-lined arrow stands for a context transition with implied return.

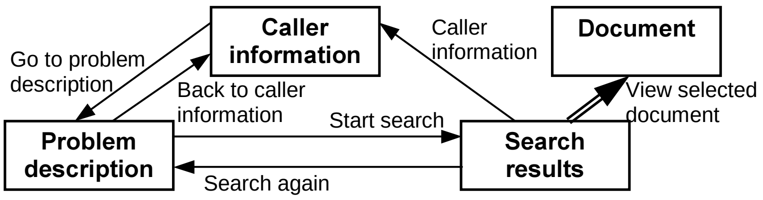
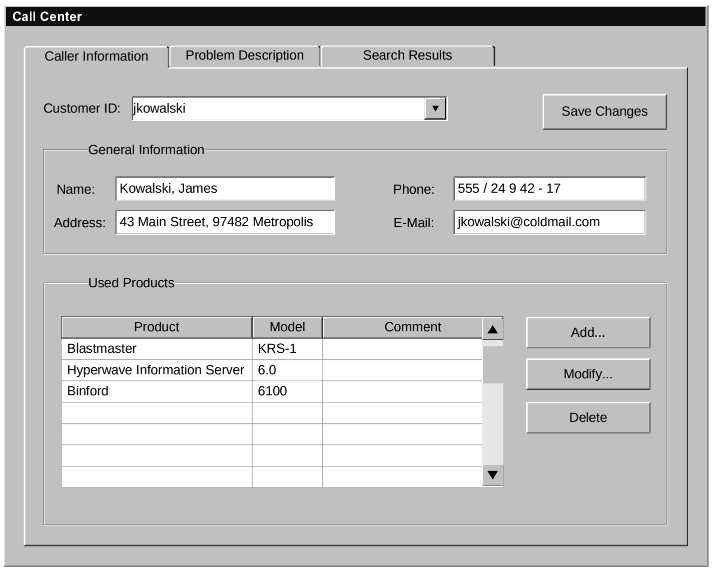


Figure 7: Partial navigation map

3.4 Visual and interaction design

The screenshot below [Figure 8] shows a possible design for the *caller information* screen in the *serve a caller* use case. As explained in [Section 2], the visual design is derived from the abstract prototype, by applying interaction patterns to particular parts of the interface. For example, the caller identification was realized using a *continuous filter* [van Welie and Trœttberg, 2000], that allows narrowing down the list of possible customer IDs dynamically, and thus providing a very efficient way of finding the desired ID, with efficiency being one of the usability goals for that role. Other patterns used in this screen include *unambiguous format*, that prevents a user from entering illegal characters into the input fields, or *grid layout* for the arrangement of the elements on the screen.



Customer ID: jkowalski

Name:	Kowalski, James	Phone:	555 / 24 9 42 - 17
Address:	43 Main Street, 97482 Metropolis	E-Mail:	jkowalski@coldmail.com

Product	Model	Comment
Blastmaster	KRS-1	
Hyperwave Information Server	6.0	
Binford	6100	

Figure 8: Visual design for the caller information interaction context

4 Summary and Future Work

The purpose of this research described in this paper was twofold: firstly, to employ established UI design methodology to KMS. For demonstration purposes we chose the example of an inbound call center where we found several user roles. Every role has special purposes and characteristics and therefore – ideally – shall receive a *special* user interface. We showed how to develop the user interface for the call center agent in one particular case. The main advantage of using this methodology for developing a user interface is, that it provides a way to assure the user interface matches the users' needs and requirements. It also serves as a tool to validate the problem domain has been understood, as well as to develop a simple shared language with the customer early in the implementation process.

References

[Constantine et al, 2000]: Constantine, Larry L. et al: From Abstraction to Realization in User Interface Designs: Abstract Prototypes based on Canonical Abstract Components. Working Paper, Constantine & Lockwood, Ltd., 2000.

[Constantine and Lockwood, 2001]: Constantine, Larry L. and Lockwood Lucy A.D.: Structure and Style in Use Cases for User Interface Design. From: van Harmelen, Mark (Editor): Object Modeling and User Interface Design: Designing Interactive Systems. Addison-Wesley, 2001.

[ForUse] ForUse – Constantine & Lockwood Ltd. home page for practitioners of usage-centered design. Questions and answers about usage-centered design. http://www.foruse.com/Questions.htm

[Hansen et al, 1999] Hansen, Morten T, and Nohia, Nitin, and Tierney, Thomas. What‘s your Strategy for Managing Knowledge, Harvard Business Review, 1999.

[Hyperwave] Hyperwave. http://www.hyperwave.com

[van Welie and Trœttberg, 2000]. van Welie, Martijn and Trœttberg, Hallvard: Interaction Patterns in User Interfaces. In: 7th Pattern Languages of Programs Conference, 13-16 August, Allerton Park Monticello, Illinois, USA, 2000.

[van Welie, 2001]: van Welie, Martijn. Task-based User Interface Design. PhD Thesis, Vrije Universiteit Amsterdam, April 2001.

[Zack, 1999] Zack, Michael H. Knowledge Directions, Journal of Institute for Knowledge Management, IBM, 1999.