CHIC - A pluggable solution for community help in context

Gunnar Stevens
Fraunhofer-FIT
St. Augustin, Germany
gunnar.stevens@fit.fraunhofer.de

Torben Wiedenhöfer
University of Siegen
Siegen, Germany
torben.wiedenhoefer@uni-siegen.de

ABSTRACT

Online "Helps" must capture the problem of de-contextualisation. In the literature the following three methods are presented to bridge the gap between the user's problem situation and the provided help: using methods of the information retrieval, computer mediated communication and techniques of context awareness.

Focusing on professional help only is a drawback most help systems research have today. In order to overcome these shortcomings, this paper presents how the different help methods can be combined with the concepts of community based help systems by using Wikis. We will argue that the next big step is to integrate Wikis into the applications so that there is a more seamless transition between the use context and using the Wiki as a Help System.

In order to prove our concept, we designed a CHiC (Community Help in Context) prototype based on Eclipse, and use it in a rich client for a Groupware-System.

Author Keywords

Help Systems, Wiki, Context Awareness, Virtual Communi-ties, Knowledge Management

INTRODUCTION

Applications are increasingly becoming more complex and the appropriation of such systems represents an immense effort for the users. Today, reducing the appropriation cost is a huge economical challenge. Several techniques have been developed to support the appropriation [17]. Of course, a good design should provide an intuitive handling of the software, and this is the best way to support the appropriation. However, from personal experience, this doesn't always happen. In particular, complex systems have to deal with this problem. As a consequence, "*most user-friendly software systems* are *not always easy to use*" [8].

In such cases, experimenting with the systems is a commonly known strategy that helps the user to better run the system. However, in order not to reinvent the wheel, it is very helpful to benefit from the knowledge of others. In particular, asking someone else and studying the documentation may play an important role in the appropriation process. Help systems are especially designed as "meta-artifacts" to provide information about a particular artifact.

Following the argumentation [18, 24], the general fundamental problem of appropriation support, and in particular online help systems, is with the problem of de-contextualization and re-contextualization. De-contextualization comes from the fact that the specific problem situation of the user is unknown to the designer while designing the help texts. Therefore, it is necessary for the user to recontextualize the help to their specific situation.

First we provided a survey of the online help research. Then we reinterpreted the literature under the perspective of the problem with de-contextualization and presented several different approaches that exist in dealing with this problem. This leads to a categorizing scheme which can be applied to study the situation of Community based Online Help Systems of today. This analysis demonstrates the existence of a blind spot that we want to fill in with our Community Help in Context (CHiC) concept. Finally, we conclude this analysis with presenting a general scheme for this system and a design study on its implementation, which was later evaluated.

APPROACHES TO ONLINE HELP

"Online help is a topic, procedural or reference information delivered through computer software. Most Help Systems are designed to give assistance in the use of a software application or operating system, but can also be used to present information on a broad range of subjects." Wikipedia 15/3/2005

Online help systems is its own topic in Human Computer Research [7, 8, 10, 19, 22, 26]. The beginning phases of online help systems were modeled similar to their paper counterparts to the degree that they were in many cases were simply a user manual with limited rudimentary search and browse capabilities [24]. Beyond that, there were some interesting developments which have an impact on the design of online help:

NordiCHI 2006: Changing Roles, 14-18 October 2006, Oslo, Norway

Copyright 2006 ACM ISBN 1-59593-325-5/06/0010…$5.00

References and Information Retrieval Based Systems

References should support the users by leading them to the right place in the user manual. Examples of techniques for content organization include: multiple indexes, alternative vocabularies, alternative entry points, and embedded hyperlinks. Typical tools for the user are the different types of search engines. In general, references are a special field of the information retrieval research.

As references are special types of information systems, they also inherit these problems of decontextualization. In particular, a major drawback of references is that one needs the special skills to appropriately use them e.g. the user must guess which subject is used for his problem. This is a well-known problem in informational retrieval research.

Knowledge Based Help Systems

Knowledge based help systems try to provide help with respect to the specific needs and knowledge of the user. Like other artificial intelligence (AI) based approaches, it is grounded in techniques of user modeling. The user model is given by an ontology that predefines the user's aspects which can be represented in the system, and a database containing the 'measuring data' that classifies both the user and a set of interference rules. The 'knowledge' will be used by the systems to individualize parts of the help systems, e.g. the reference system or the content of the online manual. Yet, it can also be used to provide pro-active hints for help.

The approaches can be differentiated in adaptive and adaptable systems. The adaptive system dynamically adapts itself to the user data, the usage data, and the environment. On the other hand, the adaptable system changes only under explicit user control [12, 14].

The major problem of artificial intelligence, particularly for knowledge-based help systems, is that it can only be applied to well-known micro-worlds [11]. As a consequence, designers must possess a considerable body of knowledge about the (problem) domain and the interference rules. However, there is always the possibility that the user model is wrong or contains false, outdated, and/or inadequate information. This leads to the system supplying inappropriate or confusing answers, or that the user is interrupted in an untimely manner. As knowledge-based help systems are heavy-weight solutions, it is quite impossible for the user to understand the system and therefore make a workaround in order to obtain the right answer to their question [24].

Context help

Context help can also use AI, but the distinctive feature is that context help is settled up on the concept of direct manipulation [21]. The main idea is to present the help in the context of the manipulated object or to give some information on an object of interest. The feature is not to use the object of interest to capture the help information only, but to use the object's place to present the help information.

Examples of context help are Macintosh Balloon or Microsoft Tooltip. On the Windows systems, it is also common to use the F1 key to provide context sensitive information to the selected object.

The context help uses the heuristic that the information to an object of interest has something to do with the problem situation of the user. This heuristic can also use algorithm of a knowledge based approach.

Tutorials and Animated Demonstrations

The concept of Tutorials is to familiarize the user with the application by common use scenarios. Tutorials are set up in a way where the scenarios are done in a step-by-step pre-planned fashion. How the information is presented can vary. Animated demonstrations, such as real-time instantiations of computer-based procedures [16] use a different medium than textual instructions. Some comparison studies were conducted to analyze the effects of different media to the user's learning progress. Bétrancourt and Tversky reviewing this research concluded that computer animation is not a panacea in itself, but it can improve users' performance and attitude under certain circumstances [5].

Frequently asked Question (FAQ)

FAQs are based on the common problem patterns experienced when a novice appropriates a new application. Aggregating the individual problems create a pattern that leads to the phenomena that the support must answer frequently asked question. Historically, Frequently Asked Question (FAQ) lists arose in the Usenet, but they are also common in the WWW [15]. FAQs are a growing memory, saving the experience of the persons, who help other users, when they run in trouble.

However, in practice, FAQ's are generally listed in an ordering system that is not clear within the list of questions. Their power to quickly supply relevant answers, without having to resort to more elaborate help systems were tempered by the fact that, as they grew in size, their ease of use became increasingly more frustrating.

Computer Mediated Communication (CMC)

Another way in utilizing the computer in order to provide help is not to see the computer as an information system, but as a communication medium. Under this perspective, email, chat, instant messaging and other applications can be a part of a help system strategy. Here the system provides users with problem-solving assistance and peer support in a collaborative computer-mediated community. Systems can be differentiated in whether they allow more synchronous (chat, instant messaging) or more asynchronous communication (Newsgroups, Bulletin Boards), and whether it is a one to one, a one to many, or a many to many communication.

Although this help is quite effective and highly contextualized, professional support is expensive and also, the approach is faulty. Firstly, communication by definition is transient. The communication process has to start from the beginning each time when a problem occurs. Secondly, in a given problem situation, there will not always be an expert available for help.

To overcome these problems, researchers have build special software systems (primarily for call centers) which routes a question to an immediately available specialist and attach a question with gathered costumer information.

AnswerGarden

The principle behind the AnswerGarden is to provide a CMC infrastructure and combine this concept with a hierarchal FAQ [1, 2, 3]. In other words, AnswerGarden addresses the problem, store the data of FAQ's, and provide a way to access and organize this data inside the communication problem.

The communication process in AnswerGarden is structured as follows: A user goes through the FAQ's and in the case where he does not find an appropriate response to his question uses the AnswerGarden to send his questions to an expert. These questions and answers will be stored in the AnswerGarden data base arranged in a hierarchical structure. This structure can be adapted in order to fit the needs of the user. So the help memory can grow in a structured way.

DEALING WITH THE GENERAL PROBLEM OF DE-CONTEXTUALIZATION

A major problem with electronic help systems is the fact that the help offered by the systems does not take into account the actual use context. Furthermore, this support does not completely fit into the personal breakdown situation [24]. Therefore, it is difficult for the designers of help system to fulfill the requirements in a way that the help systems "really help the user to understand what the application means and what the users need to do with it" [23]. We call this the problem of de-contextualization.

One can find different ways in dealing with trying to categorize the approaches mentioned in this paper. In analyzing the Help Systems' approaches, we found three different strategies:

Document centric approaches

Creating help texts in various document formats is the oldest approach of Help Systems. At the beginning online help means only to have an electronic copy of the paper based help manual. Yet, by enhancing the text with a hyper structure and providing tools for information retrieval, the problem of de-contextualization can be adequately dealt with. Such techniques allow the users to contextualize the help by creating individual and situated strategies to retrieve the information. Improvement of such approaches will primarily be driven by the progress in the information retrieval research.

Computer Mediated Communication Approaches

Sociology, and in particular, Ethnomethodology, demonstrates that in communication processes, humans own the capability to de-contextualize and re-contextualize the communication in respect to the particular situation.

The concept of Help Systems using Computer-mediated communication is based on this human faculty. In opposite to the document centric approach, it does not provide help texts but provides primarily an infrastructure that supports human-to-human communication. With the provided infrastructure, a help desk uses the communication channel to supply feedback in a natural and thus being able to clarify and contextualize the problem situation.

Approaches of context-aware help-systems providing help in the use context

These approaches are based on watching the user's behavior and his preferences. We can define three variants in dealing with the problem of contextualization: active help in offering the user active help in certain use-situations; adaptive help in reconsidering the history of usage and from this basis adapting the help-systems by using AI algorithms; and context-sensitive help in offering help due to the up-to-date context.

COMMUNITY BASED HELP SYSTEM

A general problem of all classical help systems is the apparent dichotomy between the design and usage of the system that leads to a time, space, and cultural gulf between the designers helping the user and the users asking for help. To reduce this gap, the concept of "helping people to help themselves" arose.

The strength behind community based systems is their fast response to things that are not foreseen at the time of design and their collective memories that they provide to all potential users who test the artifacts. On the other hand, these groups, (especially Newsgroups) have the problem of storing their memory in an unstructured way. In this case, the architecture of Wiki systems are more suited to organize them.

Community based Help Systems have the advantage of being a collective memory that is growing as it is used. However, today community based systems are not well integrated into the application and the use context. In particular, this is true for Wiki based solutions. Through the loosely coupling with the application, these systems have not the advantages of built-in help systems. The help systems that are built-in can be directly accessed with a "single click" inside the use context. This increases the usability of such help systems and it also allows these systems that capture and adapt the help information to the context. Both direct access and adaptation used in the context will reduce the cost of using such a helping systems.

	Produced by Professionals only	**Produced (also) by Community/Peers**
Document-centric Help Systems	Online manuals or hypertext help-systems	Wikis, Archived CMC oriented help-systems
Computer Mediated Communication Help Systems	Call Centers, Professional Help Desks	Mailing Listing, Newsgroups, Bulletin Boards, Answer Garden
Use context aware Help System	Active Help Systems, Adaptive Help, Context-sensitive Help	

Table 1 Classification of the different approaches of help systems

Based on our survey, we can summarize the state of the art in a matrix, categorising the help system approaches of today (cf. Table 1).

The idea of filling the gap in the matrix will put together the best of both worlds, creating an infrastructure that allows the people to help each other, but reducing the cost to use this infrastructure in a particular working context, e.g. getting community help with a single click while writing a paper with a word processor. In the following section a general architecture for such a system is introduced along with its design study to show how such a system can be realized. We used a Wiki as a backend system and integrated it into the Help Systems of the Eclipse Framework.

EXISTING SOLUTION USING WIKI AS AN HELP SYSTEM

*"A **Wiki** or **wiki** is a web application that allows users to add content, as on an Internet forum, but also allows anyone to edit the content. […] A wiki enables documents to be written collectively in a simple markup language using a web browser." Wikipedia 3/4/2005*

By definition a Wiki is a content management tool that allows collective writing. It can be used to write online help manuals with a hyper-structure to navigate through the help systems in an individual way. In some sense, the prominent Wikipedia (a general propose encyclopedia) is an example for an "online help system" for everything.

When the content of a Wiki will be focused to a special software system or a use context for such a software system, the Wiki becomes a community based online manual. This means that on a technical level, the common known Wikis can be used as a classical community based online help system with an integrated hyper structure. It is not surprising that some open source projects use a Wiki to provide such online documentation for their software systems (e.g. Linuxquestion[1] or Shareaza[2] use a Wiki base FAQ List).

Are their also combination of CMC help approaches and Wiki?

In a strong sense, CMC and Wiki systems support a different type of cooperation. CMC systems, like chat or newsgroups, are dialog or thread oriented. The user benefits from the direct response of the opposite partner. In opposite Wiki pages based on the idea of collaborative writing, editing one document. Roughly speaking, the difference between a chat and a Wiki is that a Wiki allows changing the text of someone else.

However in a broader sense, one can consider that Wikis are a special kind of asynchronous communication. In particular, some Wikis have the functionality that allows the discussion of the content of a Wiki page in a special area (e.g. Wikipedia, SnipSnap). This is a step that combines the concept shared material with communication tools. Another approach to bring CMC and Wikis together is to adapt the idea of AnswerGarden. A prototypical implementation of this idea was conducted by Prestipino [20]. His prototype is a community system for travel information. The novelty of this system is that it integrates a discussion forum into a Wiki. To support the integration, every new sub-forum received its own page in the Wiki system. The prototype also allows linking Wiki pages inside a discussion thread, which they later automatically received a reference to the citation in the discussion forum.

Our research into existing Wiki based solutions does not contain any hints that combine context help approaches with community based approaches.

COMMUNITY HELP IN CONTEXT

The mentioned gap in the help system approaches should be filled in by the concept of Community Help (CHiC). The general architecture of a CHiC-System is broken down into three generic modules AIM, CAM and CBHS. Although this architecture can be applied to every application system, we believe that the re-designed efforts made are smaller for component based systems, such as Eclipse (see below) with a centralized help system.

To demonstrate the usefulness and the usability of our ChiC concept we have designed a reference implementation for Eclipse. In order to fulfill the requirements in applying CHiC to every application system and adapting any community help system, we broke down the general

[1] See: http://wiki.linuxquestions.org/wiki/Common_Questions_and_Misconceptions (visit: 3/4/2005)

[2] See: http://wiki.shareaza.com/static/FAQ (visit: 3/4/2005)

architecture into three generic modules AIM, CAM and CBHS.

The CHiC-Architecture

- **AIM**

The Application Integration Module (AIM) integrates the CHiC into an existing application. The user interacts with the CHiC system through the AIM. The AIM allows asking for help inside the use context. It provides a "single-click" access to the net based CBHS-System or other help texts found on the internet (like relevant newsgroup entries).

- **CBHS**

A Community Based Help System (CBHS) can be a traditional Community System enhanced with special functionality that is needed to provide context-sensitive help.

- **CAM**

The Context-Aware Adaptation Module mediates between the AIM and the CBHS. It adapts the CBHS (e. g. by providing the right entrance Point) depending on the actual use context and/or the use history.

Designing in the context of Eclipse

We chose Eclipse because of its component based architecture and its complex and adaptable help system. In our case, we do not think about Eclipse only as an appropriate Java IDE, but we believe that Eclipse will be more and more popular for complex and high adaptable rich client applications, e.g. like IBM Workplace [13]. We use the implementation of the CHiC solution to enhance the Eclipse based Rich Client for the BSCW Groupware system with context sensitive help. Due to the fact that most parts of the application don't support context sensitive help, we have tried to find good hooks for context help via reflection techniques [9].

The Eclipse Help System

Before we explain how we integrated the CHiC-System into the Eclipse framework, we gave a brief description of the Eclipse Help, describing in particular, the context sensitive and the dynamic aspects of the help system.

Eclipse provides a highly sophisticated and adaptable help system. This help system supplies a text search capable of finding information by keyword and a context-sensitive help that describes the particular function, views, or buttons that you are working with. You can interact with the help system on the workbench using the help view or an external help browser window.

Context sensitive help

Context-sensitive help allows you to receive help while working through a task. By bringing the focus to the interface widget in question and pressing the F1 key, the user is able to access the context sensitive help. A help view will appear at the right side of the Eclipse workbench and display a list of related topics. Context-sensitive help is contributed by creating a help context extension and associating that context to a view. The *setHelp* method registered the SWT Control with a *help identifier* which must be explicitly specified by the application developers. This help identifier is associated with related help documents.

Dynamic creation of context help

In addition to providing help entries statically, the Eclipse help system has a dynamic help mechanism. This help mechasim assist pages in integrating beyond the ones which were just done by ordinary help extension mechanism. For example, one can integrate dynamic web pages with the help of this concept.

In order to integrate this new mechanism into the existing framework, the Eclipse framework developers have reused the adapter concept of Eclipse [4]. We extended the original adapter concept, introducing an algorithm to calculate context identifiers automatically in order to provide context sensitive help on a part which was not foreseen by the application developer.

The architecture of the implementation

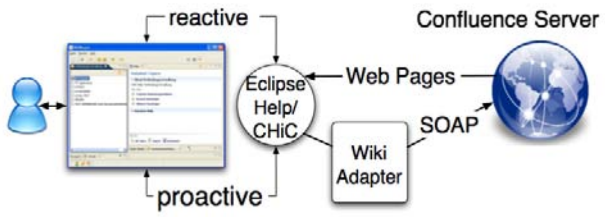
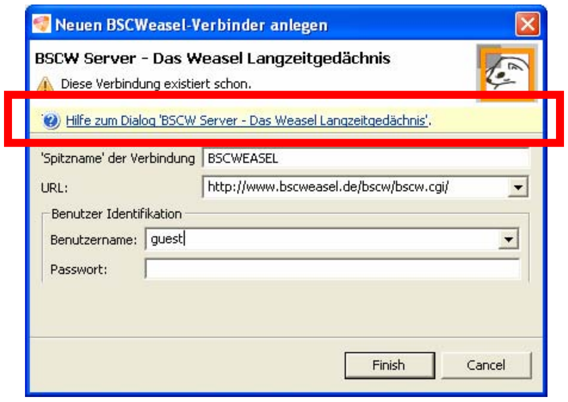


Figure 1 Realization of the general parts of the CHiC architecture

Figure 1 shows the realization of the parts of the CHiC concept using the Eclipse Framework and the Atlassian Confluence Wiki System. The next section describes the realization of the different parts of our solution.

The AIM realization

The goal of the AIM is to calculate an entrance point into the CHBS system.



Neuen BSCWeasel-Verbinder anlegen

BSCW Server - Das Weasel Langzeitgedächnis

Diese Verbindung existiert schon.

Hilfe zum Dialog 'BSCW Server - Das Weasel Langzeitgedächnis'.

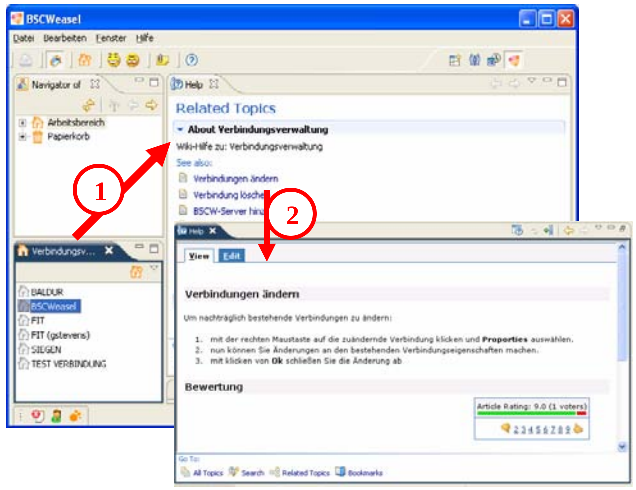
'Spitzname' der Verbindung	BSCWEASEL
URL:	http://www.bscweasel.de/bscw/bscw.cgi/

Benutzer Identifikation

Benutzername:	guest
Passwort:	

Figure 2 Integrating context specific links to jump to the right place in the Wiki Systems.

We decided to extend the standard Eclipse Help framework in order to reach a smooth integration into the ordinary help of an application. The context sensitive help in Eclipse can distinguish between a re-active and a pro-active mode (see Figure 1). In re-active mode, the user interfaces have a link that characterizes the help specific context. This link can be set by the designer manually or can be automatically interwoven by the CHiC system code inspection and runtime information on the rich client application. The different strategies to calculate a hook to the help system are explained later. **Figure 2** shows an example of a user interface that was enhanced with a link to a context help. In the pro-active mode, the Eclipse help framework observes the state of the rich client application and adapts the help text in respect to the observed state. The Eclipse framework allows adapting this functionality by implementing so called IContextProviders. The help framework will invoke the IContextProvider, when the state of the application has been changed and ask the provider to return a set of help entries.



Verbindungen ändern

Um nachträglich bestehende Verbindungen zu ändern:

1. mit der rechten Maustaste auf die zuändernde Verbindung klicken und **Properties** auswählen.
2. nun können Sie Änderungen an den bestehenden Verbindungseigenschaften machen.
3. mit klicken von **Ok** schließen Sie die Änderung ab

Figure 3 A change in the focus triggers to adapt the help entries (1). A click on the entry will open the Wiki page in the internal web browser (2)

We write our own IContextProvider that uses the algorithm mentioned later to inspect the object which is the focus of the user. The calculated context identifier was used to retrieve the associated help entries from the CBHS. The name of the entries together with the URL of the entries will be added into the context sensitive help of Eclipse (see Figure 3-1).

A click on a help entry opens the corresponding Wiki page inside the help system of Eclipse (see Figure 3-2). In addition, special links are added into the help view of Eclipse to add new help entries to the given context. The tight integration of CHiC into the ordinary help system of Eclipse leads to a user experience that is similar to the use of the ordinary help (see the Evaluation section).

The CBHS realization

To combine the context-sensitive help capability with a community concept, we chose the Atlassian Confluence Wiki to meet the "helping users to help themselves" philosophy. The primary application domain for the Confluence was Knowledge Management within organizations. Confluence has a Web Service interface, which allows to retrieve the content as well as to adapt the Wiki System via remote procedure calls.

A feature of Confluence is the hierarchical structuring of the Wiki pages. This means that pages can have child pages. This was a very important feature in using Confluence as a CBHS for CHiC. We map a context identifier to Wiki page using the context identifier as the name of the page. The help entries of a context were dynamically added as children of this specific Wiki page.

We use the Web Service API to retrieve the Wiki pages as well as to create new help entries. With the help of these Web Services it was possible to use an *off-the-shelf* version of the Wiki system to meet the requirements.

The CAM realization

The standard mechanism to link "help texts" to a particular context is to add context identifiers in the source code of an application. Normally, the user interface widgets have a special parameter which is set to the context identifier. The bottleneck in this approach is the frequency at which designers forget to write help texts or integrate the context identifier in the right place in the source code. The lack of this realization is due to theoretical and an economical reasoning. The theoretical reasoning is the situated actions of use are difficult to predict on design time [24]. The economical reasoning includes writing help texts and adding extra context identifiers in the source code cost an extra effort.

With the help of the CHiC concept we have solved the later problem, providing a solution which will not require developers to manually add special context identifiers. We implemented an algorithm which automatically calculates a context identifier.

Automatic context capturing

In order to design this algorithm, we conducted in the first step a user study. This study involves for the users to describe in a snapshot, what they would consider as an object of interest. We were not interested in the meaning that users have assigned to the object. We were only interested in the structure of the pixel that has been labelled as an object, and in what methods people have applied to identify these pixels as an object.
In the next step, we analysed the source code of the Eclipse framework. In particular we studied for whether we would find patterns that can be used to identify this kind of user perceived objects automatically. We then wrote an algorithm that uses the reflection functionality as well as

other additional runtime information to calculate an appropriate context identifier.
We identified at least three requirements for the quality of automatic calculated context identifier:

1. **The algorithm should produce an identifier that is stable in time.**
2. **The user should be able to reproduce the condition that leads to a particular context identifier**
3. **The algorithm should capture semantic meaningful use contexts. In the best case, only a particular use context is related to the context identifier.**

The first requirement is quite obvious. The traditional context sensitive help of Eclipse fulfils this requirement which demands that the application developer has to integrate a unique help identifier to their source code. However, one of the goals for CHiC was to provide context sensitive help even on places, which were not planned by the application developers. In this situation, the requirement was not so easy to fulfil.

The second requirement came from the fact that the CHiC was a dynamic community based and interactive help system. This allowed the user to extend the content to a particular context given by the calculated context identifier. The extension can serve not only as a help text, but also as a comment or question to a community. In particularly, in the last case, the user should be notified about changes in the help pages to this context. This can be done by the different notification mechanism provided by the backend Wiki system. Nevertheless, the user should be able to start his application, reproduce the situation where the problem occurs, and open CHiC to see his and the comments of the community.

The third requirement came from the fact that the source code and the runtime information provide a large amount of information about the application state. However, major part of this information is only technically motivated and has no meaning to users in its own context. For example, the user interface shown in Figure 3 has about 60 graphical controls which can automatically evaluate. Yet some of them are more or less meaningless to the user, e.g. the composite controls which were only introduced to layout the other controls.

In the light of these requirements, we have developed some strategies to calculate an identifier.

Using the existing help identifier of the application

The first strategy in calculating a context identifier is to use the context help identifier that is provided by the application developers. In the case of Eclipse, one can use the *WorkbenchHelp* class of the Eclipse Help System to retrieve this information. These help identifiers fulfil the three requirements mentioned earlier.

As a rule of thumb, a help id is an indicator that a developer uses to determine whether a special use context needs attention. .

The only problem is that in practise the application developers do not integrate such information into their code. This can be due to today's time pressure in the software development. Developers do not have the time or budget for writing help pages. In a traditional help system, "help identifiers" doesn't make any sense when the corresponding help pages do not exist. When concepts like CHiC becomes more popular and allow the separation of specifying a help context and writing a help page, these situations may change in future. Nevertheless it makes sense to think about strategies which deduce a context identifier without this information. These will reduce the coding effort and also open the opportunities to offer help in situations which are not planned by the application developers.

Using information from the Eclipse extension mechanism

A good candidate for generating an identifier without the special help ids that's provided by the application developers is to use the information from the Eclipse extension mechanism. In order to manage the different extensions, almost all extensions must have a unique id. In the Eclipse Framework one can find different ways to capture the identifier of an interesting extension. For example, we added an Adapter in the global Eclipse *AdapterManager* to calculate id of the active view.

In particular, extension of the user interface such as Toolbar extensions (views but also perspectives) are candidates that fulfil the three requirements mentioned before.

A drawback to this strategy is that not every interesting point is given by an extension. This means that there is no extension identifier which can be used to reference the context.

Using the object identifier

In object oriented systems like Eclipse (as written in Java), each object has a unique object identifier. However, this identifier cannot be used as a reference to the CBHS as it does not fulfil the first requirement. In most cases, the object id is only transient information and different instances of the same application will produce a different object id for semantically identical objects (like a menu button in the toolbar).

Using class name information

A different case is the class of an object. This information will fulfil the first requirement (modulo versioning), but many class names do not convey any application specific information (in particular the Class *java.lang.Object* is too general). Nevertheless, through the existing coding practice, in some situations the class name is a good candidate for a context identifier.

For example, the complete context menu approach of the *JFace* part of Eclipse depends on the class inspection mechanism. Eclipse *JFace* is a Model-View-Control (MVC) architecture that recognizes tables, trees or other standard user interfaces. In *JFace* the standard UI (like a table) is connected to the application specific model through a so called content providers and content labels. The developer of the model should express the application specific semantic of the model (also in the class and interfaces) which are implemented by the model object. Third party developer can extend the context menu of the view of a model using the Eclipse extension mechanism. We can use these coding conventions of classes and interfaces for our purpose in order to calculate an identifier for the domain specific model.

Another example of the strategy is shown in Figure 2. Here, the algorithm retrieves the title of the original dialogue in order to build the name of the link. In addition, the algorithm also retrieves the actual class name of the dialogue using the polymorphic feature of Java. It uses this class name[3] to generate that context identifier.

Using a mix of information

Although the mentioned strategies cover many of our cases, some interesting points will not be captured by these strategies. For examples a menu in the toolbar isn't covered when it was native integrated by an application not using the Eclipse extension mechanism. In such situations, we need an alternate way of calculating a context identifier. Here the algorithm should use a mixture of information to fulfil the requirements of a good context capturing mechanism. In the case of a menu in a toolbox, one can use the icon and the title of the menu to generate an identifier. The design rational of these strategies is that both items (icon and title) are the signs that convey the application specific meaning. However, this is one of the points that we have to prove in future.

EVALUATION

Innovative design concepts are difficult to evaluate, in particular, community based solutions. Hard facts can only be provided when the system is successfully appropriated by both the single user and the community as a whole. However, these are only long term processes, so at this stage of realisation, we can only attain temporary results about the concept. Therefore, a triangulation approach was used in studying the two issues separately:

- Exploring the current state of the use of the existing help systems to determine explicit resistance against ideas of community based help approaches
- Explicating the lived orderliness of the person's practical engagement with the objects of de-sign [6]

The first point was evaluated by a quantitative questionnaire survey, addressing the use of traditional help systems. We chose a heterogeneous set of participants that included public servants, employees from producing industries, and students of different disciplines (n=58).

The survey shows that 74% of the users experienced problems within the last month, but overall 86% of the subjects have rarely or never used traditional help systems. Both numbers indicate that people avoid using help systems, even when searching for help. This dissonance is grounded by the results that 62% of the people complain about receiving appropriate help information in general when using traditional help systems. Seventy-one percent demanded to find related help systems only with relatively high cost, and furthermore, 68% of the subjects were unsatisfied with the content of help items they received in order to solve their problems. Cognitive barriers show the fact those fifty-eight percent experienced problems by understanding the help information. These statistics reflect the dissatisfaction of the current standard help, and points out the necessity for alternative help concepts.

The second point was analysed by a standard oriented walkthrough task of BSCWeasel[4], a rich client for BSCW Groupware System. The users have to solve typical tasks using the groupware system. The focus in our study was not on the usability problems of the groupware system, but on observing the practical engagement of the users with the CHiC solution as an integrated part of the BSCW groupware client. Initial help texts were provided by the developers of the BSCWeasel, who uses CHiC instead of ordinary help systems in implementing a context sensitive help for the BSCWeasel.

After the study, informal interviews with the participants were conducted in order to better understand the usage patterns. The study ran between 1 and 1 ½ hours per user, and included 6 participants (2 developers of the original BSCW developers – not included in the design of BSCWeasel and CHiC – and 4 regular BSCW users).

During this evaluation the large-scale integration of CHiC into the context of BSCWeasel, re-using the existing help system of Eclipse, became a critical point. We observe that the common scheme of interpretation [6] using CHiC was influenced by the background expectation of traditional built-in help systems. At the one side the smooth integration

[3] In the case of dialogues, we make use of the common Eclipse coding practice that developers write a new class, when they add a new dialog in their application. This leads to an individualized class name, e.g. in the case of Figure 2 the absolute name of the class was *de.uni_siegen.fb5.bscw.siteview.views.BSCWSiteCreationPage*. This work practice of the developers can be used to refer to the use context of the dialog.

[4] Cf.: http://www.bscweasel.de

increases the usability of the help system, but on the other side it also clarifies that it was difficult to see the community aspects on the surface at a first glance. A specific case was a participant who uses the CHiC systems as a traditional help system in the usability study. In the interview afterwards, discussing the idea of integrating community based help system into the application context, he gave us the recommendation that a Wiki based help system would be a useful tool and it may be a good idea to combine this with the online help systems – not realizing that he used such a system five minutes ago. Another participant responded to our question of why he was having problems seeing the community aspects of CHiC. He missed the direct response of an acting community and indications such as registered users, comments or notes about the composition of the help item etc., based on his experience with community systems. During the task oriented walkthrough, it was realized that three of the participants did not feel inclined to begin using the help system. This was deeply rooted from former experiences with help systems. Frustration and unsuccessfulness with traditional help systems were expressed in of some the feedback we had already received from their questionnaire survey. These key factors can be attributed to what influenced the use of the CHiC-System.

All in all, the observed appropriation leads to a negative result in respect of the community issues. Although the questionnaire survey as well as our interviews in the usability study indicates that there is not a resistance against help contributions coming from other users (e.g. 92% percent of the users who participated in the questionnaire survey relied on texts written by other users), the observation shows that there is a lack of motivation for users to collaborate actively with the help system. For instance, people assume a working help system for the money they paid for in an application and are most likely not willing to invest in any work that helps to improve the system. In the case where people would not find any help items for their problem, they would be unmotivated to send a request to the community and wait for a new help text. Instead, they require an immediate response.

However, at the same time, the use of the CHiC prototype in the usability study indicate that through a smooth integration of CHiC into the application and quality of the automatic context capturing algorithm, CHiC can be used as an alternative to the ordinary context help system. The participants who have searched for help and used the help system were satisfied with the quick and effective suggestions they received, and therefore gave favourable ratings to the selection of help items according to their use context.

In summary, the evaluation demonstrates that at this stage the CHiC prototype was accepted as an alternative to the existing built-in context help of Eclipse, but it supports the community based help idea only at a technical level.

In its practical use, the *Scheme of Interpretation* [6], CHiC was utilized mostly as an ordinary built-in help system instead of as a community based communication tool.

CONCLUSION

The research has demonstrated that the appropriation of computer systems can benefit from community based help systems [1]. Nevertheless, such community systems are not well integrated into the application systems. In particular, they do not take the application context into account.

This paper focused on the technical aspects of the problem of integrating the concept of community help into the application context. On a technical level, our CHiC implementation provides several benefits over the traditional context sensitive help system:

- With the help of the automatic calculation of a context identifier, the solution can be applied for legacy applications.
- New applications also benefit from the automatic calculation because it reduces the cost adding context identifier into the source code manually.
- Context specific help texts can be directly added by usability experts when they evaluate an application. Help texts can be added, even when the application is deployed.

On the other side, evaluating the use of CHiC in the BSCWeasel client also demonstrates that CHiC isn't yet appropriated by the community of users writing their own help entries. Therefore, it is an open question whether or not the community help culture will emerged in the future. Nevertheless, two different scenarios can be identified of how such a culture can be developed:

- Foster community activities using concepts of community engineering

Based on the observation that the chosen realisation does not lead to a self-running community, additional means can be implemented to promote the active participation of the user community. A technical means in promoting an active use of the help system is to redesign the system in a way such that it provides affordances for active participation, making the community aspects more visible. Beside the technical means, non-technical aspects of community engineering can also be implemented. An example would be to hire professional users to create an artificial community in order to overcome the initial critical mass problems, and/or providing an incentive for active users such as receiving a free BSCWeasel T-shirt for making five help text suggestions.

- Organic grass root appropriation of the community aspects

The second scenario is based on the observation that CHiC in itself provides several benefits over the traditional help solutions. This can lead to producers using CHiC or CHiC like concepts as a infrastructure for a professional help system, although they may not care to take an active participation. Yet, since the community functionality of CHiC is free, users can take possession of community aspects. In such a scenario, the community based help serves only as a latent opportunity provided by the CHiC technical features. Nevertheless, this gives the opportunity that it can be manifested into an organic grass root appropriation the community features of CHiC an active user community.

The first scenario seems to be more realistic when CHiC or CHiC like concepts are adopted by companies that a have large project budgets and a sizeable user community.

The later seems to be more adequate for small and medium size projects where the designers are not intimidated by active user participation, and users feel responsible for their product.

REFERENCES

1. Ackerman, M., Malone T. (1996) *Answer Garden 2: merging organizational memory with collaborative help*. Proc. of the conf. on Office information systems: ACM Press. 97-105.
2. Ackerman, M., Malone, T. (1990): *Answer Garden: A Tool for Growing Organizational Memory*. *ACM SIGOIS Bulletin*: ACM Press. 31-39.
3. Ackerman, M.S., McDonald, D. (2000) *Collaborative Support for Informal Information in Collective Memory Systems.* Information Systems Frontiers. 2(3-4). 333-347.
4. Beck, K. Gamma, E. (2004): *Contributing to Eclipse: Principles, Patterns, and Plugins.* Addison Wesley.
5. Bétrancourt, M. & Tversky, B. (2000): *Effect of computer animation on users'performance: a review*. Le travail Humain, 63(4). 311-330.
6. Crabtree, Andy (2004): *Taking Technomethodology Seriously: Hybrid Change in the Ethnomethodology-Design Relationship.* European Journal of Information Systems. 13(3). 195-209.
7. Carroll, J.M. (1997) *Minimalism beyond the Nurnberg Funnel*, MIT Press, Cambridge
8. Covi, L., Ackerman, M. (1995) *Such easy-to-use systems: How organizations shape the design and use of online help systems*. Proc. of conf. on Organizational computing systems. NY: ACM Press. 280-288.
9. Dourish, P. (1992) *Computational Reflection and CSCW Design*. Rank Xerox EuroPARC, EPC-92-102, Cambridge, UK, 1992
10. Dix, J.; Finlay, E.; Abowd, G.; Beale, Ru., eds. (1998): *Human-Computer Interaction*. 2nd Ed., Prentice Hall.
11. Dryfus, H. (1972): *What computers can't do*. New York, NY: Harper and Row.
12. Fischer, G. (2001) *User modeling in human-computer interaction*. UMUAI 11(1-2), NY: Springer. 65-86.
13. Grama H.; Attenborough, K et al. (2004): *IBM Workplace Client Technology (Rich Client Edition) Technical Overview*. Redbooks.
14. Kobsa, A., and Wahlster, W., eds. (1989). *User Models in Dialog Systems*. NY: Springer.
15. Ngambi, D. (2002): *Pre-empting User Questions through Anticipating – Data Mining FAQ lists*. Proceedings of SAICSIT. (Port Elizabeth, South Africa) NY: ACM Press. 101-109
16. Palmiter, S. & Elkerton, J. (1991). *An evaluation of animated demonstrations for learning computer-based tasks*. Proc. of CHI'91. ACM Press. 257 – 263.
17. Pipek, V.; Wulf, V. (2003): *Pruning the Answer Garden: Knowledge Sharing in Maintenance Engineering*. Proc. of the ECSCW: Kluwer. 1-20.
18. Pipek, Volkmar (2005): *From Tailoring to appropration support: Negotiating groupware usage*. PhD Thesis, University of Oulu, 2005.
19. Preece, J.; Rogers, Y.; Sharp, H.; Benyon, D.; Holland, S.; Carey, T. eds. (1994): *Human- Computer Interaction*. Addison Wesley.
20. Prestipino, M (2004): *Supporting Collaborative Information Spaces for Tourists Conference*. Proc. of, Mensch und Computer 2004. Oldenbourg Verlag. 209 - 219
21. Shneiderman, B. (1983): *Direct manipulation: A step beyond programming languages*. IEEE Computer, 16(8), 57-69.
22. Shneiderman, B. (1997): *Designing the user interface: strategies for effective human-computer interaction*. Addison-Wesley.
23. Silveira, M., Sieckenius de Souza, C., Barbosa, S. (2001): *Semiotic engineering contributions for designing online help systems*. SIGDOC'01. NY: ACM Press. 31- 38.
24. Suchman, L. (1987): *Plans and situated actions. The problem of human--machine communication*. Cambridge University Press.
25. Turk, K., Nichols, M. (1996): *Online Help Systems: Technological Evolution or Revolution?* Proc. of the 14th conf. on Systems documentation. NY: ACM Press. 239 – 242.
26. Wulf, V. (1999): *"Let's see your Search-Tool!" - Collaborative use of Tailored Artifacts in Groupware*. Proceedings of GROUP '99, NY: ACM-Press. 50 – 60