Adapting to the Level of Experience of the User in Mixed-Initiative Web Self-Service Applications

Mehmet H. Göker
Kaidara Software
330 Distel Circle, Suite 150
Los Altos, CA 94022
mgoker@kaidara.com

Abstract. Web self service applications deliver technical support to a diverse group of typically anonymous end users. Such applications have to be able to adapt themselves to the level of experience of the user and deliver consistent problems solutions. Since most users are not uniquely identifiable, creating a long term user model is not feasible. The knowledge in the application has to be extremely easy to create, update and maintain by a team of subject matter experts that are distributed world-wide and are not experienced in knowledge modeling. We describe an approach to build a user adaptive mixed initiative web self service application that can operate under these conditions and illustrate the ideas by analyzing an application.

1. Motivation

Web self-service applications provide customer support for technical equipment in a very cost-efficient manner. These applications have to be able to support a high number of users from different backgrounds and with varying levels of experience and consistently deliver correct solutions to problems. The user interface and the interaction modalities have to be able to support both novice and expert users. At the same time, the information in these systems and the user interface has to be easy to create, update and maintain.

Personalization techniques provide means to support users from different backgrounds and with varying preferences with a single application. Similar to human beings, a user adaptive or personalized system acquires and stores the characteristics or preferences of users in a user model and adapts the manner in which it acquires, selects and presents information accordingly (Langley (1997), c.f. Göker and Smyth (2001)).

Web self-service, troubleshooting and diagnosis applications are built to support users to find solutions to their problems as efficiently and effectively as possible. For such applications, the user’s personal preferences regarding information display and interaction modalities play a secondary role. The primary goal is to understand what the failure is, determine the root cause of the failure, and find a solution to the existing technical problem as efficiently and effectively as possible. To achieve this goal, a troubleshooting system will need to obtain information needed to diagnose the problem from the user (and potentially other systems). Depending on the level of
experience of the user, the manner in which this information will be provided and can
be acquired will vary and the system will have to adapt.

Adapting to the experience of the user is something done routinely for developing
intelligent tutoring systems (c.f. Brusilowsky 1999). Some approaches used in
tutoring systems require pre-processing or chunking the information into modules that
can be sequenced or adapted according to the user’s experience. This method can also
be transferred to diagnosis problems (Brusilowsky and Cooper 1999). However, the
overhead in creating and maintaining a complex knowledge representation poses a
serious problem with respect to the long term maintainability of such an application.
Especially in an application where cases are entered by authors and subject matter
experts that are not experienced in knowledge modeling, the representation has to be
extremely straightforward and unambiguous.

Alternatively some approaches adapt themselves to the user’s level of experience
by modifying the user interface, available interaction options and the presented
information (e.g. Ardissono and Goy (2000); Fesemmaier, Ricci, Schaumlechner,
Wöber, and Zanella (2003)). These approaches treat the experience of the user as part
of a user model. However, building a user model requires the ability to uniquely
identify a user. This is not necessarily possible for web self service applications.

We are interested in developing a web self-service application that allows users
from different backgrounds to search for solutions to their technical problems. While
some parts of the application will require user authentication, others will not. This
prevents us from uniquely identifying users and creating reliable user models. The
system is required to adapt itself to the level of experience of the user, acquire the
necessary and sufficient information to diagnose a failure, and consistently retrieve
the same solution for the same technical problem. The knowledge in the application
has to be extremely easy to create, update and maintain by a team of subject matter
experts that are distributed world-wide and are not experienced in knowledge
modeling. This requires the representation to be very simple, easy to understand and
unambiguous.

In the following, we describe and analyze the TAC Case collection\(^1\) of Cisco
systems as an application which adapts itself to the level of knowledge of the end
user. The application provides network engineers support while troubleshooting Cisco
equipment. Currently, the “wireless” section of the application is publicly available
(\[\text{http://www.cisco.com/public/support/wireless_launch.html}\]) while other sections are
password protected and only available to employees of Cisco customers. The analysis
is based on the Kaidara Advisor\(^2\) Software and the dialogue operators described for
the Adaptive Place Advisor (Göker and Thompson (2000)).

\section{Application Development with Kaidara Advisor}

Kaidara Advisor is a development environment for building structural Case-Based
Decision Support systems. Since it is a \textit{structural} CBR development environment
(Bergmann et.al (1999)), the resulting systems are based on an application specific

---

\^1 Technical Assistance Center (TAC) Case Collection, © Cisco Systems, \url{http://www.cisco.com}
\^2 © Kaidara Software, \url{http://www.kaidara.com}
domain model that contains the vocabulary (attributes, values, and their structure) for the application area. Cases and queries are a structured set of attribute-value pairs that are based on a representation derived from the domain model and stored as records in database tables. Kaidara Advisor provides the following three modes of interaction with the end user:

- **Free Text**: The end users can type in their query using free text. This query is parsed, ambiguities clarified, and used to create a structured query for the CBR Engine.
- **Guided Search**: Using information gain, Kaidara Advisor applications guide the end users by asking them to specify values for the most informative attributes, thereby guiding them towards a solution in an efficient and effective manner.
- **Expert Search**: For experienced users, Kaidara Advisor applications provide the ability to enter a query in a structured form that exposes part of the domain model. The user can also specify attribute weighting factors at query time.

Especially for troubleshooting and diagnosis applications of complex, technical equipment, we have found that novice users prefer to enter their problem description by using the free text entry modality or to be guided in a step-by-step fashion. Expert users, on the other hand, may feel slowed down by the free text or guided modalities. Since they do not get intimidated when asked to characterize their problem by using the necessary and sufficient number of values in a structured form, they prefer the expert mode. However, even the most knowledgeable user will appreciate some guidance occasionally, and every novice will gradually learn. By seamlessly merging and linking the three modalities described above, user adaptive mixed initiative systems can be built. At the same time, the users still have the ability to use their preferred search method directly.

A typical interaction with a Kaidara system can start off with a natural language query. This query is parsed, ambiguities removed by means of clarification questions (which value is meant, which attribute is the value related to etc.) and an initial structured query sent to the CBR engine. If the number of results returned by the retrieval engine exceeds a pre-set threshold, the system will automatically switch to guided mode, asking questions and guiding the user to a solution. If the users are not satisfied with the result set, they can modify the query directly by accessing the expert search mode.

### 3. User Adaptive, Mixed Initiative Web Self Service

As described above, we are interested in developing a web self-service application that is able to deliver consistent solutions to technical problems, independent of the level of experience of the user. The goal of such a system can be described as:

*Acquire enough information to constrain the query in such a way that only a specified number of solutions above a given similarity threshold remain.*
To achieve this, the system has to adapt itself to the user’s level of experience and how he/she chooses to provide information. Assuming that the interaction modalities described above are available, a typical TAC Case Collection session with a novice user would proceed as shown in Table 1.

**Table 1: Sample interaction with a mixed-initiative diagnosis application (Novice)**

<table>
<thead>
<tr>
<th>Interaction</th>
<th>Dialogue Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User</strong></td>
<td>It looks as if I am connected to the network but I cannot send any data.</td>
<td>PROVIDE_CONSTRAIN</td>
</tr>
<tr>
<td><strong>System</strong></td>
<td>Do you mean your system is associated but cannot pass any traffic?</td>
<td>CLARIFY</td>
</tr>
<tr>
<td><strong>User</strong></td>
<td>Yes</td>
<td>PROVIDE_CONSTRAIN</td>
</tr>
<tr>
<td><strong>System</strong></td>
<td>Which products are you working with?</td>
<td>ASK_CONSTRAIN</td>
</tr>
<tr>
<td><strong>User</strong></td>
<td>Access Point and a wireless card</td>
<td>PROVIDE_CONSTRAIN</td>
</tr>
<tr>
<td><strong>System</strong></td>
<td>What is the state of your radio connection?</td>
<td>ASK_CONSTRAIN</td>
</tr>
<tr>
<td><strong>User</strong></td>
<td>Authenticated and Associated</td>
<td>PROVIDE_CONSTRAIN</td>
</tr>
<tr>
<td><strong>System</strong></td>
<td>Most similar cases are displayed in a table.</td>
<td>RECOMMEND_ITEM</td>
</tr>
<tr>
<td><strong>User</strong></td>
<td>Selects a solution from the result table</td>
<td>ACCEPT_ITEM</td>
</tr>
</tbody>
</table>

The interaction is started by the user, but depending on the data provided, the system takes initiative to acquire more information. The user can modify the information he has provided before (back, edit), ignore the system’s requests and provide responses to alternate questions (skip), ask the system to display what it has found so far (submit), or provide the answer to the current question and continue (next) (see Figures 1, 2). A user can also choose to switch modalities directly by selecting ‘Free

---

3 The example is taken from the public part of the Cisco TAC Case Collection (http://www.cisco.com/public/support/wireless_launch.html, © Cisco Systems, 2003). Due to the constant addition of cases as well as domain model, synonym and similarity metric upgrades, the retrieval results and interaction steps will vary over time.
text search', ‘Guided Search’, ‘Advanced Search’ from the tabs on the top of the application. As such, the application is a mixed initiative system (c.f. Allen (1999), McSherry(2003)) where both sides can take control of the interaction when needed, where the user can provide volunteer information and recapture the control of the system at any time, and the system can select questions to guide the user.

The problem described by the user in Table 1 is caused by the client not being able to obtain a dynamic IP address from the DHCP server (Figure 4). The description of the novice is (as usual) rather symptom oriented and does not provide any relevant technical information. A novice also uses a terminology that differs from the one the expert (and the system) would use (Figure 1). This requires the system to parse and interpret the language of the user based on the synonyms and matching rules it has available.

Assuming the same symptoms and the same root cause, an interaction with an expert may look the way shown in Table 2.

---

4 After analyzing the session logs of the application, ‘Advanced Search’ was removed from the user interface for standard users and replaced by ‘View All Solutions’. As expected, only very few users had been actually using the advanced search capability in this complex application domain.
### Table 2: Sample interaction with a mixed-initiative diagnosis application (Expert)

<table>
<thead>
<tr>
<th>Interaction</th>
<th>Dialogue Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User</strong></td>
<td>Even though the client is associated and authenticated, it cannot pass traffic. The hardware is a AP 1100 and a patch antenna</td>
<td>PROVIDE_CONSTRAIN</td>
</tr>
<tr>
<td><strong>System</strong></td>
<td>Problem Observed = Associate but cannot pass traffic; Wireless Stage =Authenticated and Associated; Product = AP 1100; Antenna = Patch Antenna;</td>
<td>VERIFY_CONSTRAIN&lt;sup&gt;5&lt;/sup&gt;</td>
</tr>
<tr>
<td><strong>User</strong></td>
<td>Accept</td>
<td>ACCEPT_CONSTRAIN</td>
</tr>
<tr>
<td><strong>System</strong></td>
<td>Most similar cases are displayed in a table.</td>
<td>RECOMMEND_ITEM</td>
</tr>
<tr>
<td><strong>User</strong></td>
<td>Selects a solution from the result table</td>
<td>ACCEPT_ITEM</td>
</tr>
</tbody>
</table>

While the problem and the result are identical, the interaction is different. The system extracts more (and different) information from the user’s query, does not need

---

<sup>5</sup> **VERIFY_CONSTRAIN** and **ACCEPT_CONSTRAIN** are two new dialogue operators that were not present in the Adaptive Place Advisor. **VERIFY_CONSTRAIN** is a system operator that ensures that the system’s interpretation of the user’s natural language input is correct. **ACCEPT_CONSTRAIN** is the corresponding user operator to accept the system’s interpretation.
to interpret as much, and, after verifying that it understood correctly, proceeds directly to the result display. Obviously, real life situations will be between these two extremes and novice users will learn the vocabulary of the system over time.

The TAC Case collection differs from other user adaptive applications in that it does not utilize a user model, only uses one simple knowledge representation scheme and only one user interface to adapt to varying levels of user experience. By using a mixture of natural language understanding, structural case-based reasoning, and information gain computations, the application ensures the user arrives at a solution in the fastest possible manner. In a troubleshooting environment, this will always be the goal and to the benefit of the users – independent of their level of experience or communication skills. The system can be thought of as a user adaptive system that is indifferent to the user. It does not really ‘know’ the level of experience of the user at any given time. Its only goal is to be able to obtain as much information as needed to determine what the root cause of the problem is. If a very inexperienced user copies an error message into the natural language text field and the system is able to identify

![Figure 3: Result table of the Cisco TAC Case Collection](image-url)
the problem and its solution immediately, it will have achieved its goal. And so will the novice who has a solution to his problem.

Figure 4: Case Details from the Cisco TAC Case Collection

The benefit of using a conversational, mixed initiative approach in terms of the ability to respond to user’s needs independent of their domain experience has also been reported in Gupta, Aha and Sandhu (2002). The main difference between the two approaches stems from the representational differences of the underlying CBR systems. The application described here has been built using a structural CBR system that allows easy authoring and maintenance of cases by separating the domain model (with the associated hierarchical representation of attributes and values), the data (flat, individual records in a database table) and the interaction modality of the user with the application (generated by means of information gain computation for guided search). We believe that a representation methodology such as the one described in Gupta et al. (2002) may be difficult to maintain given the complexity of the application domain, the high number of cases that are stored in the system and the number of concurrently working authors and subject matter experts.
4. Summary and Future Work

We have described a mixed initiative diagnosis application that adapts itself to the level of experience of the user. While typical user-adaptive or personalized systems utilize a user model, modularized knowledge or configurable user interfaces to achieve this task, the described application makes use of a combination of natural language understanding and information gain based guided search to achieve this task. As such, it only uses one knowledge representation, a single user interface and no user model. By adapting the way it acquires information, it ensures that users arrive at a solution to their problem as fast as possible, independent of their level of experience. Since the web self-service environment cannot guarantee the availability of uniquely identifiable users, the system cannot create a long term user model. However, giving the users the ability to register themselves and store certain characteristics of their environment seems like a reasonable thing to do as a future step.

The application has been live on Cisco’s web-site since April 2003. Currently we are collecting feedback and are planning to enhance the application accordingly.

Acknowledgements:
The author would like to thank David Aha, Pádraig Cunninham, Hector Muñoz-Avila and Francesco Ricci for their extremely helpful comments and suggestions on the initial version of this paper.
References


