An Avatar-Based Help System for Web-Portals

Helmut Lang[1], Christian Mosch[2], Bastian Boegel[3],
David Michel Benoit[4], and Wolfgang Minker[1]

[1] Institute of Information Technology, Ulm Univ., 89081 Ulm, Germany
[2] Communication and Information Center, Ulm Univ., 89069 Ulm, Germany
[3] Institute of Information Resource Management, Ulm Univ., 89069 Ulm, Germany
[4] Institute of Theoretical Chemistry, Ulm Univ., 89069 Ulm, Germany
{helmut.lang,christian.mosch,bastian.boegel,
david.benoit,wolfgang.minker}@uni-ulm.de

Abstract. In this paper we present an avatar-based help system for web-portals that should provide various kinds of user assistance. Along with helping users on individual elements of a web page, it is also capable to offer step-by-step guidance supporting users to complete specific tasks. Furthermore users can input free text questions in order to get additional information on related topics. Thus the avatar features a single point of reference, when the user feels the need for assistance. Different to typical systems based on dedicated help sections consisting of standalone HTML pages, help is instantly available and displayed directly at the element the user is currently working on.

Keywords: HCI; Computer Assisted Learning; Grid Computing.

1 Introduction

The recent tendency in personal computing to shift content from a local machine to the cloud has led to a new kind of interfaces. Applications running on a local machine are subsequently replaced by programs accessed by web browsers. The success of netbooks and the development of Google's chrome OS are reflecting these changes. Nevertheless web based systems often do not feature equal ease of use, as programs run locally. This is mostly due to deficiencies in current web browsers along with immature standards.

Web based help systems nowadays usually consist of a set of HTML pages that force users to switch their focus of attention from the page they are currently editing towards this dedicated help pages. Another downside of an HTML based approach is that users, especially those new to the given field, have the feeling to get lost. [16]

A web-portal currently developed in the context of the bwGRiD project [2] should overcome this problem. The portal allows users to access grid computing resources from within their web browsers. Thus we decided to implement an avatar-based help system that offers the information needed by users for the completion of certain tasks right on the web page they are currently editing.

J.A. Jacko (Ed.): Human-Computer Interaction, Part II, HCII 2011, LNCS 6762, pp. 537–546, 2011.

In addition to task specific guidance the avatar shall be able to provide further information on related concepts.

The following sections we will first introduce the architecture of the web-portal used at our site along with the implications for a help system caused by the architectural approach. Then we define the demands for the help system according to a typical use case of the portal. Finally we describe, how the avatar-based system meets those demands and provide some details on its technical realization.

2 Portal Architecture

Grid computing usually requires massive usage of command line tools. Those hard to learn interfaces increase the barrier for using the grid especially for inexperienced users [1]. The objective of the bwGRiD portal project [3] is to make grid computing more appealing to novice users by offering access to grid computing resources by means of a web-portal.

In order to provide interfaces for different applications each project partner is developing portal based applications fitted for specific target audiences. The portal developed in Ulm addresses the needs of theoretical chemists. Hence it features a graphical molecule editor allowing users to prepare input files for Gaussian [13] and NWChem [12] and enables "point and click" driven submission of the generated jobs to the grid.

Fig. 1 depicts an architectural overview of the portal in use at our site. Client communication is handled by a tomcat servlet container in combination with GridSphere [9]. GridSphere is an implementation of the JSR 168 Portlet Specification [10]. Different to servlets that return web pages as a whole, portlets are only responsible to render fractions of pages. When the portlet container (GridSphere) receives a client request, it determines all portlets included in the requested page, triggers them and integrates the returned output into a response.

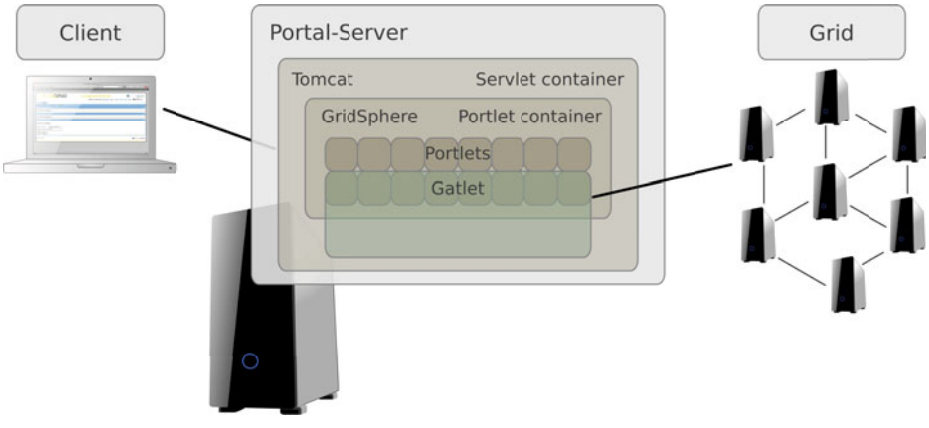

Fig. 1. Architectural overview of the portal in use at our site

The modular nature of portlets perfectly fits the idea of different sites contributing different modules for various applications. However this modularity makes developing an integrated help system for such a portal quite time and resource consuming. Also the help system has to be designed in modular way and each portlet has to provide the information needed in a self-contained manner. The system then has to combine the information available for every portlet into a unified user experience.

The portal's link to the grid is established by Gatlet [6]. Gatlet offers a set of generic portlets needed for grid access in general (e. g. to administer MyProxy credentials or to access files stored on the grid) along with an API for interaction with grid resources. Currently Gatlet features support for the Globus [8] grid-middleware but support for UNICORE [17] and gLite [7] is planned.

3 Use Cases and Requirements

A typical workflow for a chemist submitting a job to the grid would involve the following steps:

1. Login to the portal using a certificate stored in the web browser's keystore.
2. Navigate to a page containing the MyProxy portlet provided by Gatlet.
3. Use the MyProxy portlet to retrieve a proxy credential.
4. Navigate to the page containing the chemistry portlet.
5. Use the graphical editor to create a molecule.
6. Choose if you want to create an NWChem or Gaussian job and edit a set of parameters.
7. The input file generated by the portal can be edited, in case the user wants to make manual adaptations.
8. Choose the resource the job should be executed at.
9. Navigate to a page containing the job monitoring portlet.
10. Observe the status of the job until it is finished.
11. Retrieve and examine the results.

Analyzing this workflow points out some important features that an adequate help system should provide:

- There is a well-defined sequence of steps that have to be processed to complete the given task. Thus for novice users a step-by-step guide visiting all required portlets and assisting the user in taking decisions (e. g. if creating a Gaussian job is preferable over an NWChem job) is really helpful. Due to the modularity of the portal, the portlets involved during the step-by-step tutorial might be spread over different pages. Thus the system has to be aware of the portal layout and the guide has to span multiple pages.
- Users already familiar with the portal might only need help on specific parts of the interface. In this case the system should be able to provide information on individual elements.

– Interested users are also confronted with new concepts, like “credentials”. Even though understanding the relevance of proxy credentials is not essential to use the portal, users might be interested to get to know more about grid computing in general. Hence the help system should also be able to give information on continuative topics related to grid computing and chemistry. This feature is of particular relevance, since the target audience of the portal are people new to grid computing, like students.

We came to the conclusion, that these features can be implemented with an avatar-based system quite elegantly and that such a system would be superior to other approaches. The advantages of such an avatar-based framework will be discussed in the following section.

4 Features

When the avatar is idle he is displayed as a small icon along with an “Assist me” button (see Fig. 2). In order to keep the avatar from interfering with other elements on the page the user can drag him to different locations.

While trying to exploit the well known “persona effect” [11] we also wanted to keep the effect of reduced performance, achieved when the user feels observed by an animated agent [14], as low as possible. Hence the avatar does not exhibit any idle animations.

To get help from the avatar the user has to click on “Assist me”. After this the actual content of the page is shaded, a short introductory message is displayed in

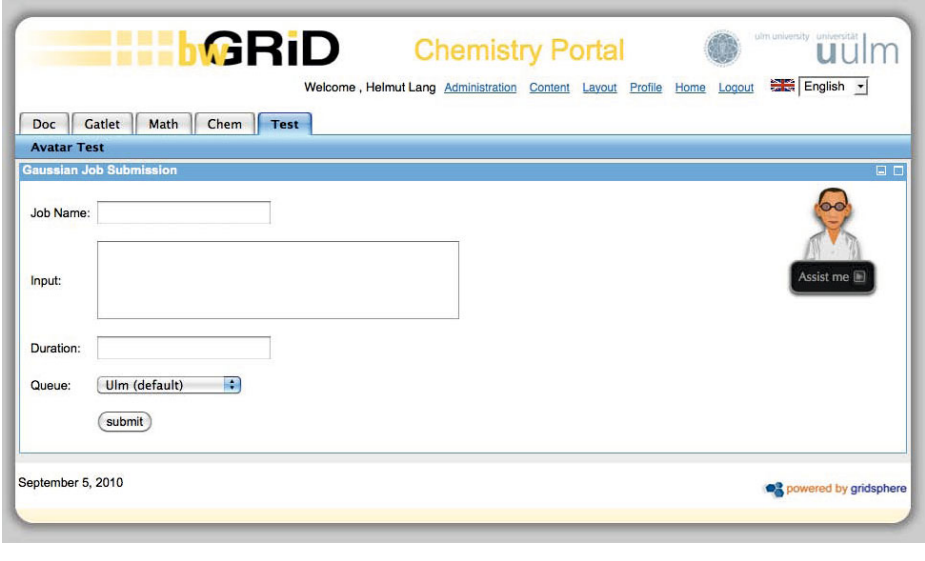

Fig. 2. The avatar in idle state

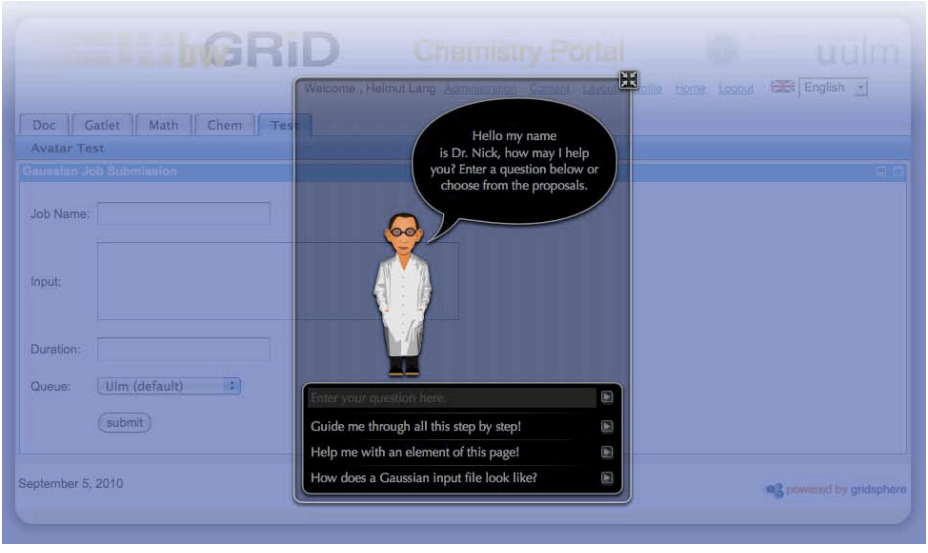

Fig. 3. The avatar after clicking "Assist me"

the avatars speech balloon and different options for further actions are presented in an interactive area below the avatar (see Fig. 3).

4.1 Element Explanation and Input Verification

The basic functionality of the avatar is to explain elements on the page and to verify the user input. To get help on an element the user chooses "Help me with an element of this page" at the greeting dialogue (see Fig. 3). Then all elements the avatar is able to explain are distinctively marked. When the user clicks on one of the marked components the avatar points at it and provides additional information (see Fig. 4).

This behavior implements the second requirement from the previous section, by providing information on elements chosen by the user. In contrast to tooltip or balloon help systems, our approach allows users to identify fields with additional information at first sight, without trying to keep the mouse as still as possible over an element. During explanation, the current input field remains editable. As a side effect text displayed in the speech balloon, may be copied and pasted. This is especially useful to provide templates for input fields of higher complexity.

Displaying the information along with the actual content reduces the cognitive load on the user, compared to dedicated help pages. [4]

Another advantage of the avatar is the capability to compensate for deficiencies of the underlying interface. In the example of Fig. 4, the user would normally have to enter a duration from 120 to 604 800 in seconds into an input field. The avatar superimposes this field with separate spin boxes for days, hours

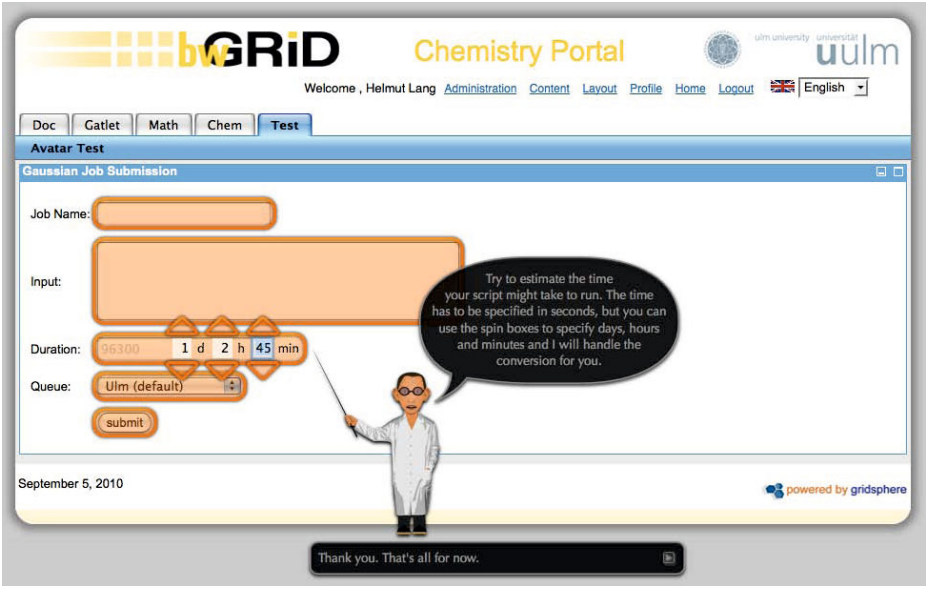

Fig. 4. The avatar assisting a user with entering a duration

and minutes, thus relieving the user from the burden of doing excessive calculations manually. Furthermore it is also possible to define minimum and maximum values and the avatar takes care that those limits are kept.

Aside from validating durations, it is possible to define a regular expression for input verification. In case of user input violating this expression, the avatar clarifies which input is actually allowed.

4.2 Step-by-Step Guidance

The avatar is able to assist users performing certain tasks utilizing step-by-step tutorials. Thus satisfying the first requirement from the previous section.

In the step-by-step scenario the avatar directs the user to the pages containing portlets that need to be visited during the step-by-step tutorial by pointing at the corresponding items in the navigational menu. On these pages all elements that need to be edited are explained sequentially. That way we have realized a typical "worked example" [15] kind of learning.

4.3 Continuative Information

The input field in Fig. 3 is used to enter questions covering continuative topics. If an answer to the entered question is found it can be presented to the user as a kind of slide show, where the avatar makes remarks on certain elements. Fig. 5 shows how the avatar explains the constituents of Gaussian input file. Next to this rather sophisticated way of presentation, it is also possible to just display some text in the speech balloon, depending on the complexity of the answer.

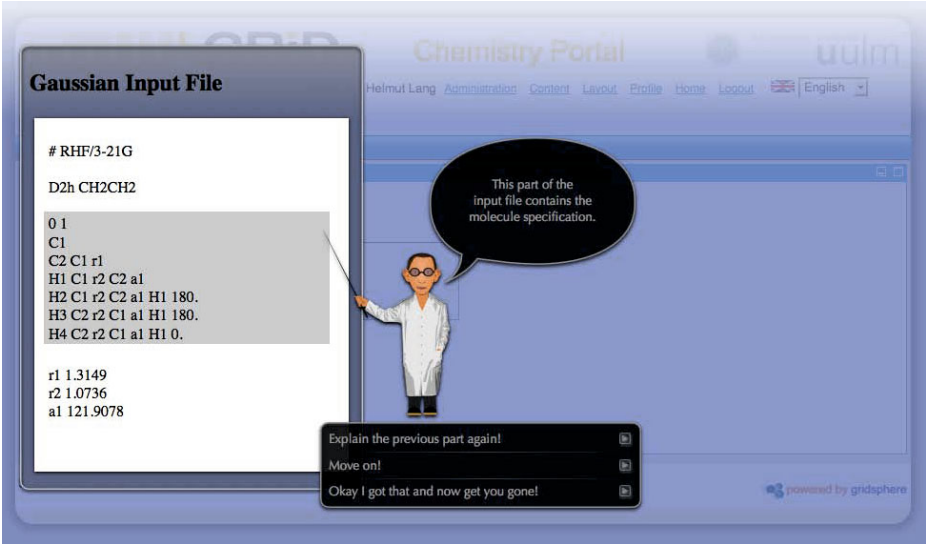

Fig. 5. The avatar explaining the constituents of a Gaussian input file

```
<?xml version="1.0" encoding="UTF-8"?>
<av:answer xmlns:av="http://uni-ulm.de/bwgrid/portal/avatar"
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://uni-ulm.de/bwgrid/portal/avatar
            ../schema/answer.xsd ">
  <av:complex>
    <av:slide title="Gaussian Input File">
      <av:content>
        # RHF/3-21G<br/><br/>
        D2h CH2CH2<br/><br/>
        <div id="molSpec">
        0 1<br/>
        C1<br/>
        C2 C1 r1<br/>
        H1 C1 r2 C2 a1<br/>
        H2 C1 r2 C2 a1 H1 180.<br/>
        H3 C2 r2 C1 a1 H1 180.<br/>
        H4 C2 r2 C1 a1 H1   0.<br/>
        </div>
        <br/>
        r1   1.3149<br/>
        r2   1.0736<br/>
        a1 121.9078<br/>
        <br/>
        <br/>
      </av:content>

      <av:explainPoint ref="molSpec" pos="EAST">
        <av:text>This part of the input file contains the molecule specification</av:text>
        <av:forward>Explain the previous part again!</av:forward>
        <av:backward>Move on!</av:backward>
      </av:explainPoint>

    </av:slide>
  </av:complex>
</av:answer>
```

Fig. 6. A reply (abbreviated for clearity) as returned for a question like "What does a Gaussian input file look like?"

Furthermore the system suggests context dependent questions, displayed as options in the greeting dialogue (see Fig. 3). That way users get an idea on how to formulate questions for the avatar and which kind of questions the avatar might be able to answer. Logging the questions asked on certain pages can be used to determine frequently asked questions and make appropriate suggestions.

5 Technical Realization

5.1 General

The client side of the avatar is completely implemented in JavaScript. We have made a significant effort in keeping it compatible to a broad range of contemporary browsers. By using JavaScript only the avatar is completely independent of plug-ins and additional software.

5.2 Element Explanation and Input Verification

The text prompted by the avatar, when the user asks for help on certain elements is embedded into the page by means of hidden divs. To generate those hidden divs a tag library was implemented that allows extending the JSP-pages used to render the portlet. The following code example shows how the component helping the user to edit durations is added to a page:

```
<%@ taglib uri="http://uni-ulm.de/bwgrid/avatar/" prefix="av" %>
<input type="text" size="20" id="duration"/>
<av:compDesc key="durHelp" ref="id:duration" type="DurationHelper"
    constraints="min=60, max=604800" next="queueDesc"/>
```

The first part imports the required tag library, the second defines the actual input field and the remainder is responsible for generating the content relevant to the avatar. The `key` attribute in the last tag refers to a value in a Java properties file. This allows to define text for multiple languages.

Next to the `DurationHelper` type a `RegexpHelper` allows specifying a regular expression as constraints for the accompanying input field.

5.3 Step-by-Step Guidance

Within portlets, steps are defined specifying a `next` attribute within a `compDesc` Element. To allow step-by-step guides to span multiple portlets, developers have to create an additional avatar.xml file, containing information about which portlet to vist, after the user is done with the respective portlet. This file is added to the list of "watched resources" in the tomcat server. Each time the file is changed a listener is triggered and the component holding the meta information about the portal is updated.

In case the user asks for a step-by-step guide, an asynchronous request is sent to a servlet on the portal server. According to the stored meta information a list of portlets, along with the pages they can be found at, is dynamically created and returned to the client. This list is afterwards kept in the browser's session storage and sequentially processed by the avatar.

5.4 Question Answering

When the user enters a question into the input field of the greetings dialogue or chooses from one of the suggestions a request to the server is triggered. To find an appropriate answer the utterance is searched for matching keywords. The following reply consists of an xml-document constituting the answer. The system distinguishes between simple and complex answers. The suitable way to present the content is chosen by the content editor. A simplified example for the structure of a reply constituting a complex answer can be seen in Fig. 6. The content of this reply would be presented by the avatar as can be seen in Fig. 5.

5.5 GridSphere Dependencies

The approach chosen for the element explanation functionality and the answering of questions does not exhibit any GridSphere dependencies and thus can be easily adapted to other portals relying on the Java servlet technology.

In contrast to this step-by-step guides are highly GridSphere dependent. This is owed to the fact that there has to be a mapping between portlets and pages they appear on. GridSphere stores this information in a dedicated file that is scanned each time the user issues a request for a step-by-step guide. Since the method layout information is stored differs between portlet containers, adapting the step-by-step functionality to other systems would require additional work.

6 Conclusion

We propose an integrated help system that is able to assist users, without having to switch their focus of attention from the page they are currently working on. Thus help is instantly available, where it is needed, when it is needed and from a unified resource. The avatar smoothly integrates several different modes of support. He is capable of providing users with step-by-step guidance, information on continuative questions and help on individual elements. Due to this variety of features the avatar is beneficial for novice as well as advanced users.

A distinctive feature of our system is the capability to automatically integrate individual portlets on different pages into a coherent step-by-step tutorial.

Apart from this step-by-step functionality that is dependent on GridSphere – the portlet container in use at our site – all components of the system can easily be ported to portals relying on the Java servlet technology. To verify the portability we installed the avatar on a portal running Liferay [5] and only minimal adaptations where needed. Consequently the system is applicable for a wide range of application.

Future work on the project will involve user studies to verify our theses on the usability of the avatar and to draw conclusions for improvements.

Acknowledgments. The bwGRiD portal project is funded by the Ministry of Science, Research and the Arts Baden-Wuerttemberg.

References

1. Altmann, A.: Direkte Manipulation: Empirische Befunde zum Einfluß der Benutzeroberfläche auf die Erlernbarkeit von Textsystemen. Zeitschrift für Arbeits- und Organisationspsychologie 31(3), 108–114 (1987)
2. bwGRiD Project, http://www.bw-grid.de/
3. bwGRiD Portal Project, http://www.bw-grid.de/portal/
4. Chandler, P., Sweller, J.: Cognitive Load Theory and the Format of Instruction. Cognition and Instruction 8(4), 293–332 (1991)
5. Enterprise Open Source Portal and Collaboration Software – Liferay.com, http://www.liferay.com/
6. Gatlet Project Page, http://gatlet.scc.kit.edu/index.php
7. gLite - Lightweight Middleware for Grid Computing, http://glite.cern.ch/
8. Globus Alliance, http://www.globus.org/
9. GridSphere Portal Framework, http://www.gridsphere.org
10. JSR-000168 Portlet Specification (Final Release), http://jcp.org/aboutJava/communityprocess/final/jsr168/index.html
11. Lester, J.C., Converse, S.A., Kahler, S.E., Barlow, S.T., Stone, B.A., Bhogal, R.S.: The Persona Effect: Affective Impact of Animated Pedagogical Agents. In: Proc. CHI 1997, pp. 359–366 (1997)
12. NWChem High-Performance Computational Chemistry Software, http://www.nwchem-sw.org
13. Official Gaussian Website, http://www.gaussian.com/
14. Rickenberg, R., Reeves, B.: The Effects of Animated Characters on Anxiety, Task Performance, and Evaluations on User Interfaces. In: Proc. CHI 2000, pp. 49–56 (2000)
15. Sweller, J., Cooper, G.A.: The Use of Worked Examples as a Substitute for Problem Solving in Learning Algebra. Cognition and Instruction 2(1), 59–89 (1985)
16. Theng, Y.L., Thimbleby, H.: Addressing Design and Usability Issues in Hypertext and on the World Wide Web by Re-Examining the "Lost in Hyperspace" Problem. Journal of Universal Computer Science 4(11), 839–855 (1998)
17. UNICORE (Uniform Interface to Computing Resources), http://www.unicore.eu/