

Portal Modules for Groupware Systems

Nils Jeners, Steffen Budweg, Wolfgang Prinz

Fraunhofer FIT

Sankt Augustin, Germany

givenname.surname@fit.fraunhofer.de

Abstract—Groupware Systems become more and more functional and complex. Often many features are integrated in a monolithic interface that should fit all needs. A new trend towards portal systems can provide flexible and user-adaptable environments. This paper therefore presents a new approach for the modularization of collaborative systems. We present the requirements analysis, the concept and implementation as well as experiences from a pilot of portal modules that were added to an existing Groupware system.

Keywords—portal modules; individualization; customization; groupware; modularization

I. INTRODUCTION

In the post-information age, we often have an audience the size of one. On the one hand mass media reaches more and more people, but on the other hand diversification grows. Individualization is going to be more and more important (cf. Negroponte [8]). Not only individualization of information sites, but also of applications: groupware is one of them.

Building Cooperation Systems that are flexible and can be easily adapted, adopted and appropriated by groupware users to fit their individual needs has been a common design goal for a long-time in CSCW research (e.g. Mackay [6], Malone et. al [7], Koch & Teege [5], Dourish [4]). Many established groupware systems provide an integrated, monolithic approach offering a one-stop system to fulfill a wide variety of user requirements and functionalities, often integrating them in one single interface for the user.

With the rise of new individualization opportunities following Web 2.0 technologies, growing numbers of end-users are getting accustomed to using Portals or Personal Information Dashboards (e.g. iGoogle¹, Netvibes² and My Yahoo³) allowing the integration and aggregation of different information sources or services.

For cooperation research and groupware design, this raises new questions how existing mature and often monolithic systems can be augmented to fit new requirements for end-user individualization and flexible integration in personal views provided e.g. by portals. In addition, the growing popularity of mobile devices like SmartPhones (e.g. iPhone, Blackberry) and Netbooks supporting ad-hoc on-the-road information access and cooperation routines furthermore often

require light-weight integration of multiple information sources as offered by portal modules and architectures.

This paper presents the realization of a new approach allowing modular integration of groupware functionalities to personal information portals. We start with a presentation of the background of this paper and the related work followed by the requirements, the concept and architecture, as well as the implementation process. Finally we present the results of the conducted evaluation and give some final conclusions.

II. BACKGROUND & RELATED WORK

Individualization is an important process. With it we transform houses into homes, spaces into places, and things into belongings. In short, we turn ordinary objects into personal properties, which we are bond to. Nobody else can do this for us. Professional designers can make things attractive, beautiful, easy to understand and easy to use. They can design something that fulfills our needs perfectly, but even so it is not something personal for us. Therefore we are all designers. We continuously manipulate our environment to better serve our needs. The design process means not necessarily to build objects, but to choose, combine and arrange them (cf. Norman [9]).

One approach for the individualization of webaccess is the use of portals. A portal in the context of this paper is a website that lets the user individualizes content as well as the look & feel. Typical examples for portals are Netvibes and iGoogle. They offer the aggregation and combination of information and functions from different sources. These services are represented in modules, which can be added, arranged and removed by the user. The user gets a personal view of the portal. He can choose the modules he wants to insert in the portal, and where they should appear. Thereby it becomes his personal portal.

The look and feel of particular portals is very similar. They often have a title bar which includes several tabs. Each tab contains several modules organized in columns and/or rows. The layout can be organized by dragging and dropping the modules.

Portal modules are also called gadgets or widgets and they have various other names. The name widget is the most common. To avoid naming confusions with GUI widgets, we use the term *portal module* or just *module*. Whereas a *Portlet* (Java Portlet Specification JSR168 [1]) is an approach to provide server-side components for only one application, our *Portal Modules* focuses on the modularity. Understandable like a mashup, that takes the components from different providers. The portal, the module and also the groupware

¹ <http://www.google.com/ig>

² <http://www.netvibes.com>

³ <http://my.yahoo.com>

system can be distributed on different providers if the interfaces are specified.

A module is a container display or a mini application which is included in a portal interface. The portal stores properties of all modules that are included and adjusted by the user. The module connects to third party servers to receive data to display. Due to security issues this often happens via a proxy server of the portal.

The idea to realize user configurable workspace for cooperative work has already been presented with TeamRooms [11]. TeamRooms enabled users to configure a workplace using a set of predefined modules (Java applets) such as notes, todo lists, sketchboards, or calendars. Each TeamRoom could be shared with other users of a group. Although this approach is similar to ours, the difference is that our approach is based on an extension of an existing groupware environment with a module based approach, while TeamRooms was aiming at a room based environment that could incorporate different special purpose applications.

A more advanced approach is presented in [12] with the FreEvolve platform that covers flexibility on three levels: software architecture, user interface, and collaboration support. Within this framework our approach is focusing on the user interface and collaboration support level, but on a different level. We do not aim at proving users a means to change the user interface in a fine grained level, but by proving building blocks that enable the user to build a new a context and activity specific user interface by combining existing modules within an existing portal environment and not within a software engineering environment. Thus we believe that our approach is more applicable and easier to use an configure for the end user.

III. REQUIREMENTS

A. User Requirements

The following bullets provide a short overview about the different steps of our requirement analysis approach:

- Initial brainstorming session with BSCW Core Developers to identify potential portal module features.
- Ranking of identified potential features by means of a user survey.
- Comparison of survey results with logfile analysis results.
- Building of Mock-Ups and paper prototypes.

According to the idea of implementing portal modules in general, a first task is to identify the specific features to be 'portalized'. What functions are interesting and needed? A brainstorming session with four BSCW developers resulted in 16 possible functions. These functions were identified by experienced Groupware developers, but consequently end-users had to be included to validate the initial feature list.

Therefore, we conducted a user survey to assess and rank the identified potential module features which was answered by 28 experienced BSCW groupware users. Every function was rated with 'Yes I want it.' (value +1), 'No I don't want it.'

(value -1) and 'I don't care.' (value 0). The sum of the values represents the need of a function and is maximal +28 for the most needed and minimal -28 for the less needed functions. Table I shows the accumulated rating of 16 ideas.

Combining the results of the expert brainstorming and user survey, the functions with a rating equal or above 9 were selected for module implementation.

These decisions were supported by the results of a general & large-scale BSCW logfile analysis conducted by Appelt [2] that yielded a similar picture of the most used functions of BSCW. Within the top ten groups of operations are also the creation of information (position 1), presentation of information (position 3), awareness features (position 4) and searching (position 8). These groups correspond to the results listed above.

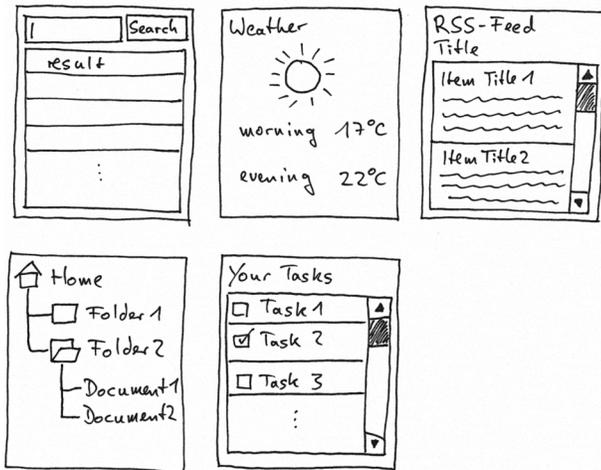
TABLE I. RESULT OF THE BSCW USER SURVEY

Function	Yes	No	I don't care	Result
Tasks and appointments	21	3	4	18
Notification of members	18	4	6	14
Upload	20	6	2	14
Folder tree structure	19	6	3	13
Status/Progress of a task	17	5	6	12
Links to an object	16	5	7	11
Events of an object	18	7	3	11
Search	16	7	5	9
Teams	13	7	8	6
Active objects	13	7	8	6
Presence of members	13	7	8	6
New workspaces	9	9	10	0
Tagcloud of a folder	7	7	14	0
Members and roles	12	13	3	-1
Blog extract	6	14	8	-8
Expectation of an object	3	13	12	-10

After the initial requirements analysis, paper prototypes of identified portal modules were created (see figure 1). This paper prototype illustrates five portal modules:

1. Search Module
2. External Information Source (e.g. weather)
3. RSS feed reader for internal and external information sources (such as BSCW awareness information, read/write events, Blog headlines)
4. Folder tree structure
5. Task-list

Figure 1. Paper prototypes



The search module on the top left searches the whole workspace of a user while the folder module displays the tree structure of a workspace. Beside there is a task list. The top center external information module is a non-BSCW module - it should demonstrate that the use of a portal is not fixed to some provider or some services and whatever works with the preferred portal can be combined. Finally, the RSS feed reader module on the top right can display both RSS feeds from BSCW (e.g. awareness information with recent changes, blog headlines) which often need authentication and external party RSS feeds (e.g. public news feeds).

B. Technical Requirements

There are three parts of module development. First the implementation of the module itself, second the embedment of the module into the portal and third the communication between the module and the groupware server.

There are many different portals and still start-up companies founding new ones while some of them already vanished. Furthermore there are desktop systems with a similar functional range (e.g. like the Windows Sidebar, Yahoo Widget Engine), but this paper only focuses on web-based systems, consciously giving a full list of all systems is not possible.

*Mashable*⁴ compared 14 portals. Only six of them offer an own API, which makes third-party development possible. Table II shows the comparison of these portals and the individual columns are explained in detail: Tabs are navigational elements that switch between several views like the current browsers provide. Every portal system provides tabs. This is a good feature to group thematically the included portal modules. The so called social features mean the ability to share modules or complete portal settings. Also the communication between users is important for this feature. Only two of the mentioned portals offer these features namely Netvibes and Pageflakes. In the sense of groupware, these features are essential. The use of RSS feeds is the naturally

portal application, therefore custom RSS feeds can be included in every portal. The layout of the mentioned portals is always organized in columns. Three to four columns is appropriate for usual desktop resolution. Netvibes introduced the most elaborate layout at the latest update (December 2008). There are not only strict columns but also layers across multiple columns. Adjustable width means the ability to customize the width of each column detached from the width of the browser window. If a portal is supported by the UWA (Ultra Widget API), it is possible to have modules across multiple portal platforms, e.g. the same module in Netvibes and also in iGoogle. The UWA of Netvibes offers a framework for module development. Beside Netvibes, the modules developed with the UWA also support other environment. To gather the amount of users is a hard task because none of the portal provider publishes this data. The numbers are estimations from the *Wired Magazin*⁵, whereas Windows Live got the user numbers of My MSN which partly became Windows Live in 2006.

TABLE II. OVERVIEW ON PORTAL PROVIDERS (CF. MASHABLE⁶ AND WIRED⁷)

Site	Tabs	Social Features	Custom RSS	Max. columns	Adjustable Width	UWA Support	Million Users
iGoogle	y	n	y	3	n	y	7
My Yahoo!	y	n	y	4	n	n	50
Netvibes	y	y	y	4	y	y	10
Pageflakes	y	y	y	4	n	n	?
WebWag	y	n	y	4	y	n	?
Windows Live	y	n	y	4	n	y	12

Currently there is no widely accepted standard or framework available except the W3C Working Draft Widget 1.0 [3]. This document is still under development. To enable module development for a distinct portal, an API is needed and there exists several APIs from the portals that applies in practice. There is the *Ultra Widget API* (UWA) of Netvibes, the *Google Gadgets API*, and some more. All these APIs follow an AJAX-inspired approach using (X)HTML with JavaScript. Beside its API every portal supporting developers can offer a container module. This container module operates as an empty module to provide the embedment in a portal. Every developer can take this container and fill it with code. Simple (X)HTML, or elaborate AJAX, embedded Adobe Flash or Microsoft Silverlight is suitable.

Table III shows the differences. Dynamic means the technique is able to load and/or reload data from a server (dynamically). Interactive means the ability to react on user

⁵ <http://www.wired.com>

⁶ <http://mashable.com/2007/06/29/personalized-homepages/>

⁷ <http://wired.com/software/webservices/news/2007/03/72999>

⁴ <http://www.mashable.com>

input and accordingly compute data on the client side. The communication with servers via HTTP or other protocols often is restricted. Therefore a method for granting access is required. A proxy server is a technique independent method. A file called `crossdomain.xml` on the server is supported by Flash and Silverlight. This file grants permission to call clients.

In addition to the above described techniques to display content, every portal module needs a technique to retrieve data otherwise it would be only a static module without any permanent importance and interest. Some of the most used techniques are RSS/Atom, REST, XML-RPC and SOAP webservices. RSS (Really Simple Syndication) and Atom are simple formats, to make data accessible to third party use. Based on a XML file stored on a server, it is possible to retrieve data just by downloading this file. Both techniques do not have a channel in the other direction. Data can be demanded but cannot be sent. A further disadvantage is the very limited format which is useful for news formats but nothing else.

REST (Representational State Transfer) is stateless transfer architecture. It is very easy to implement and to use e.g. mashups. With REST there are several URLs which represent only one function. The arguments of these specific function calls are passed via HTTP GET or POST. The result of such a call can be send as XML data.

XML-RPC (Extensible Markup Language Remote Procedure Call) is, as the name implies, a method for procedure calls between distributed systems. It uses HTTP as transport mechanism and XML to encode data. It is also a very simple system. In contrast to REST there is only one URL to call.

SOAP (Simple Object Access Protocol) is the successor of XML-RPC and therefore it has more features than XML-RPC. But it is also heavier, that means the signal to noise ratio is bigger.

TABLE III. OVERVIEW ON MODULE TECHNIQUES (CF. ADOBE⁸ AND MICROSOFT⁹)

Technique	dynamic		communication	Market Penetration at web computers
	dynamic	interactive		
(X)HTML	n	n	n	100%
AJAX	y	y	only via proxy	100%
Adobe Flash/Flex	y	y	via proxy or server permission	> 90%
Microsoft Silverlight	y	y	via proxy or server permission	< 50%

⁸ http://www.adobe.com/products/player_census/flashplayer/

⁹ <http://www.microsoft.com/presspass/press/2008/oct08/10-13Silverlight2PR.mspx>

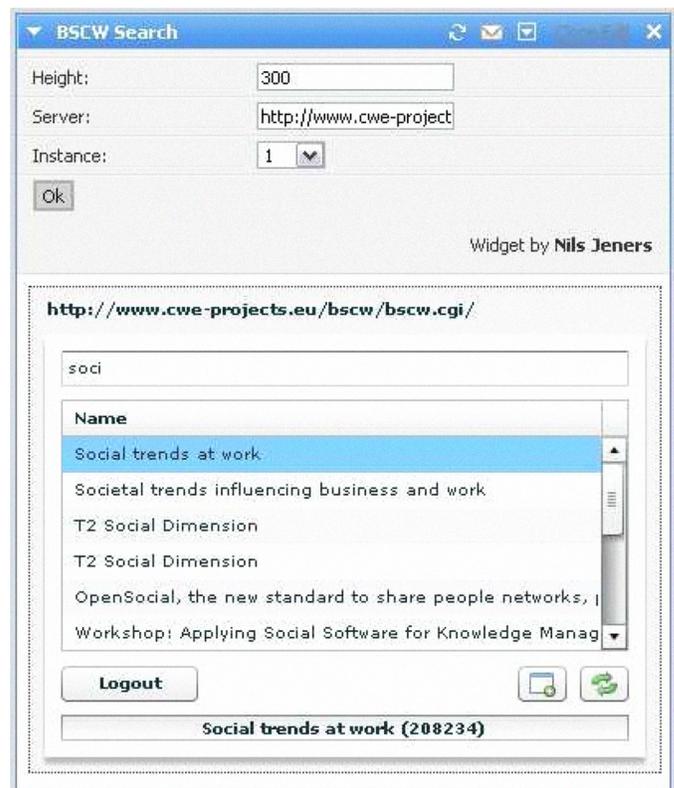
IV. CONCEPT & ARCHITECTURE

While the last chapters introduced the possible applicable techniques to portal development in general, the following present what technique we decided to use and why.

Our portal modules are implemented in *Adobe Flex* – a software development framework for *Adobe Flash*. *Flex* is specifically targeted to develop rich internet applications (RIAs). The main advantages of *Flash* are a high market penetration (e.g. compared to *Microsoft Silverlight*) and a much easier ability for cross-site scripting compared to AJAX approaches.

The UWA (Ultra Widget API) of Netvibes is used to embed modules into a portal because of one great benefit: The UWA supports more than one portal. In addition to Netvibes itself, it supports iGoogle, Live.com, Windows Vista Sidebar, Apple Dashboard, the iPhone, Opera and many more are coming soon. Therefore the user is not forced to use a specific portal.

Figure 2. Screenshot - BSCW Search module with preferences

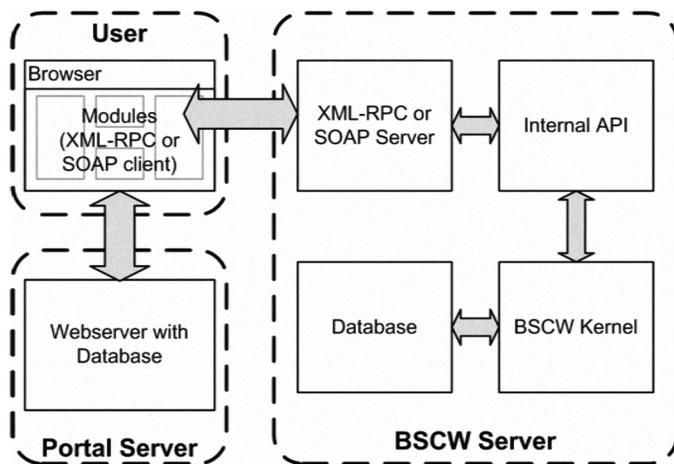


Our portal modules make use of SOAP webservices for the communication with the groupware in terms of the BSCW server. RSS is not valuable because of the missing reverse channel. REST is not valuable because BSCW does not provide it. The use of *Flex* decided between XML-RPC and SOAP because *Flex* offers a class for calling SOAP webservices and the integration of XML-RPC would have been a little harder. The BSCW Shared Workspace system is an extension of a standard web server through the server CGI Application Programming Interface. A BSCW server can be accessed either via an ordinary web browser or via XML-RPC

respectively SOAP web services. Therefore a web browser, a XML-RPC client or a SOAP client is needed. The XML-RPC interface basically provides the same functions as the human-operable web interface, whereas the SOAP webservices provide only a few functions.

In our approach the user also needs a web browser. But instead of accessing the BSCW user interface, the user contacts the portal. The portal is loaded from this server and within the portal modules possibly from several different servers (e.g. BSCW server). The portal modules are encapsulated and provide their own communication channel. In our case they call the BSCW server and interchange data. The modules have to implement an own XML-RPC or SOAP client. With the help of this clients they can contact the BSCW server in particular the according XML-RPC respectively SOAP servers. The calls from the modules are interpreted and mapped to the internal API and then computed in the BSCW Kernel.

Figure 3. Architecture



V. IMPLEMENTATION

A. UWA Netvibes Container

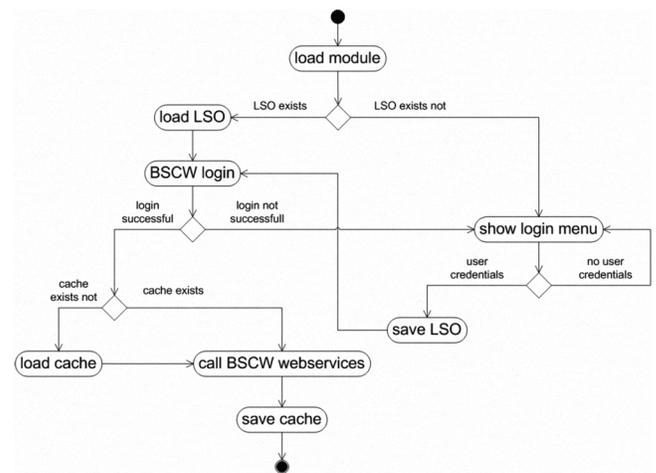
As already mentioned, every portal system offers a module container. This is a kind of a basic skeleton to start with module development. In our case, the Netvibes container module consists of a (X)HTML file including JavaScript. The (X)HTML file includes metadata (e.g. the name of the module, author, version) and further preferences that are rendered in the portal for user settings (e.g. the server URL or an object id). These settings have default values and every change made by the user is stored at the portal server, persistent attached to this specific user. We decide to use these preferences for every module settings except user credentials (username and password). The credentials are locally stored with the help of Local Shared Objects (see below), to prevent the abuse of this data. With the help of JavaScript functions (UWA) the settings that are stored on the portals server can be read and computed. In the body of the container goes everything to display in the module. The use of Flash needs only (X)HTML code in the body for the embedment and of course some JavaScript code to pass user settings from the portal to the module.

B. Adobe Flex

Flex-Applications consist of MXML, a description language based on XML, and ActionScript, for program logic. The user interface is described by MXML, which can be built with the help of the Adobe Flex Builder. The user interface objects can be combined and positioned with drag and drop. The Flex Builder implicitly helps with the layout following the Adobe Flex Interface Guidelines in providing a “magnetic” grid, where the objects snap into. This helps to have a consistent look and feel across multiple applications without much effort.

Figure 4 shows a general activity diagram of a module. It is general in terms of the initialization and retrieving data. Later computing and server connecting depend on user interactions. When a portal loads in the browser, it immediately loads every module it is containing. In the beginning, the preferences of the modules are loaded from the portal server and hand over to the individual modules. The container passes this data further to the Flex application.

Figure 4. Activity diagram of a BSCW module



The Flex application checks whether a Local Shared Object (LSO) exists or not. In this LSO data is stored at the local machine of the user like a cookie. This LSO is used to save user credentials and also to cache data. If this LSO does not successful login, the cache would be loaded if it exists. Definitely the BSCW webservice is called to update the cache and gather the latest data to work with.

The container of the module consists of a (X)HTML file with code for embedding *Flash* and *JavaScript* methods to handle the storage of the preferences. Some preferences like the server URL, object ID or height of the module are stored with the help of the UWA. Stored on the portal’s server, the data is persistent attached to one user. By contrast, the user credentials are not stored on the portal’s server. Some users do not like to store personal data on external server, so username and password are stored in a Local Shared Object (*Flash* cookie) on the user’s local machine.

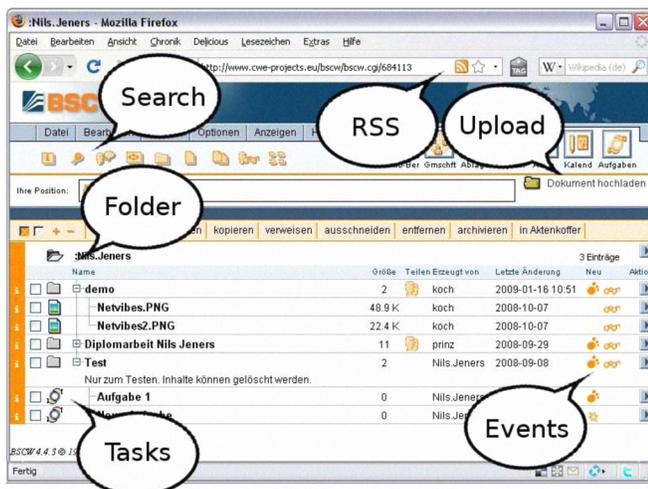
VI. COLLABORATION MODULES

Based on the requirement analysis described above we have implemented 6 different modules:

- The **search module** provides a simple search based on the document names of all documents within a specified folder as well as its subfolders. As soon as the user provides initial characters for a document name, all matching documents are listed.
- The **events module** lists all events of cooperative activities on objects within a specific folder.
- The **folder module** lists all objects sub-folders of a specific folder. Users can navigate within this hierarchy and they can open documents or folders in a new window by a double-click.
- The **task module** retrieves all cooperative tasks from the users task list in BSCW. Task can be opened in a new window for being worked upon.
- The RSS module retrieves the RSS feeds from BSCW.
- The **upload module** provides an easy way to upload local documents to the shared workspace.

Figure 5 shows the integrated web user interface of BSCW. The bubbles indicates the functions describes above (e.g. Search, RSS and Upload) within the HTML interface. Figure 5 presents the portal view of the same user showing the same functions as in Figure 4: On the top left there is a Search module, on the bottom center there is a RSS module and right aside the Upload module. Compared to the integrated web interface, the portal has nearly no menus. The functions are all visible and accessible. And the layout of the features are variable, this means the modules can be moved and arranged in a different way. Furthermore it is possible to combine modules that access different folders and workspaces of a users cooperative environment within the same portal. This is a big advantage compared to the web-interface that provides access to a single folder only. Thus it is possible to arrange information from different sources with different modules within a single context, i.e. a portal page.

Figure 5. Screenshot of the BSCW web interface



VII. EVALUATION

The development process followed an iterative process model (DIA) with three steps: Design, Implement and Analyze. The starting point is neither specified nor important. Some developers like to start with an analysis of the target user group and some start with a brainstorming session and begin with a conceptual design. Important is to have short and fast cycles to improve the prototypes stepwise. The design step always is the basis of a implementation step, whereas the analysis step always tests the implementation (prototype). The analysis leads then to a new design.

The portal module development was performed in three cycles. The single steps were brainstorming session (D), paper prototyping (I), survey (A), design decision based on the first cycle (D), first Flash prototype (I), workshop (A), design decision based on the second cycle (D), second Flash prototype (I), test phase and final user questionnaire (A).

A. Workshop

Our evaluation started with an initial one-hour workshop in which eleven users participated. This workshop was intended as a brief introduction to the concept of portal modules. After the presentation of the portal modules a focus group discussion was conducted and documented. The documented user feedback can be split in three major parts: Questions about and wishes for functions, problems and issues, and general remarks and open questions.

Questions and Wishes

Already implemented features and functions:

- An automatic refresh is provided by a portal. E.g. Netvibes reloads the modules every 20 minutes per default. But this could also be more often.
- Several instances of the same portal module (e.g. folder tree structure of different folders) is possible.

Functions those are nice to have:

- Display of the current focus of a module.
- Display of object IDs.
- Function to add modules. This is not supported by the UWA.
- A specific login module that handles the login centrally.

Problems and Issues

There was only one problem mentioned, namely the absence of a Flash Player PlugIn for a 64 bit OS. To use a Flash Player it is necessary to start a 32 bit emulator. This slows down the system performance.

General Remarks and Open Questions

The mentioned remarks were discussed in the user group, but could not be answered finally.

- Which functions are meaningful and are used?

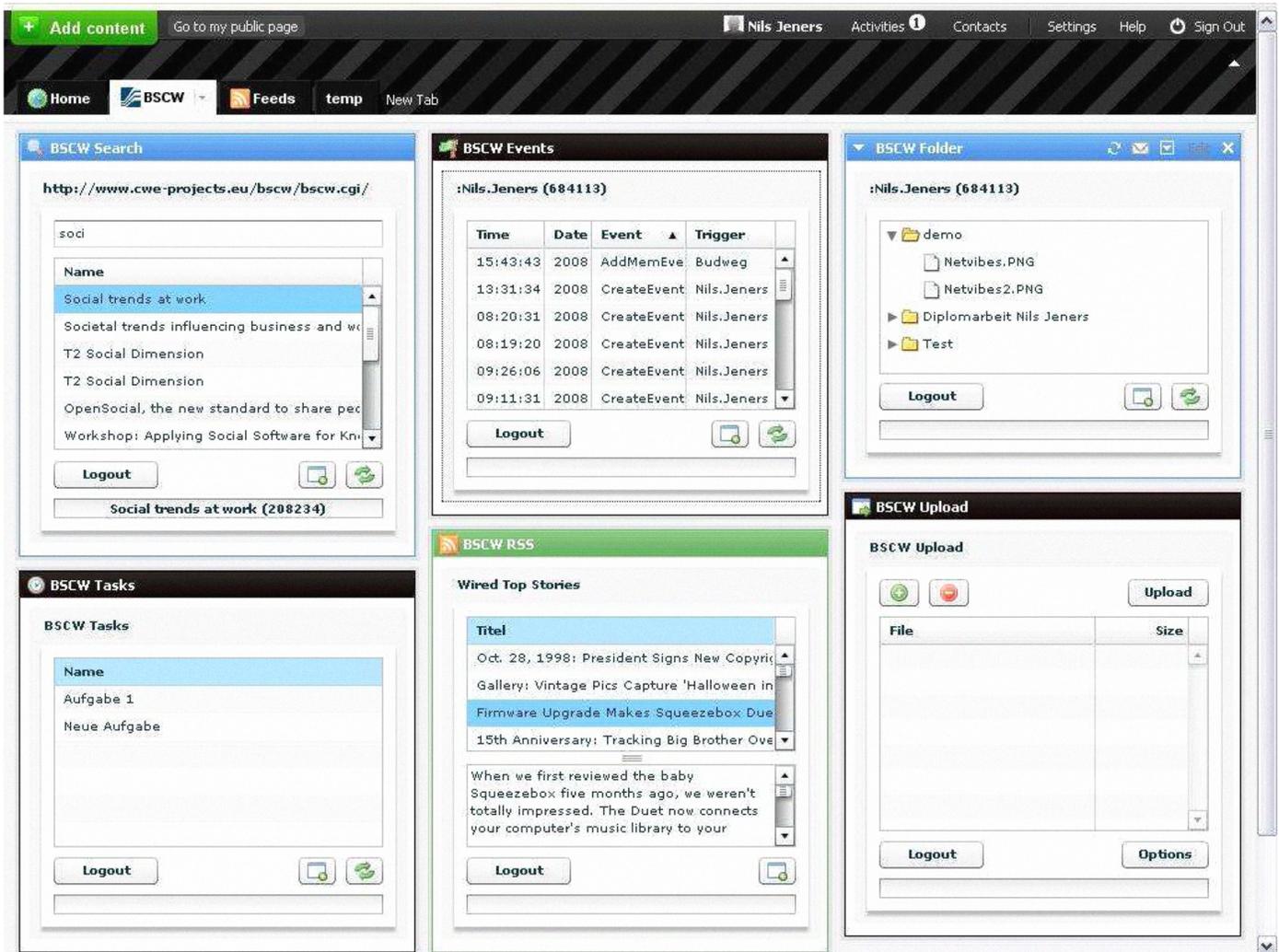
- Are portal modules an alternative for the integrated user interface of BSCW and do they have advantages?
- Which functions do users use in BSCW and which in the portal modules?

B. Test Phase

During the test phase, users got the possibility to send feedback directly from inside of the portal modules. Beside this user feedback every module reported its own usage with log files. In this log files information about the user, the time and date and the action was saved. The analysis of the logfiles from the testing phase revealed that the most used portal modules were (in decreasing usage frequency):

1. RSS-Reader
2. Folder tree structure
3. Search
4. Events
5. Upload
6. Tasks

Figure 6. Screenshot of the Netvibes portal with six BSCW portal modules



C. Final User Feedback

Subsequent to the test phase, a questionnaire asked users to think about problems, features and further comments. Below we summarized the essence of the answers.

- **Meaningful Functions.** Like already known from the test phase, the RSS-Feed, Folder, Search and Events are the most frequent used modules. But this does not mean that everybody used these modules constantly. The users only chose the modules that were meaningful for them. One user for example only chose RSS-Feeds and Folders and never touched the others. One user mentioned that the events were too raw and not specific enough.
- **Module combination and layout.** The layout of single modules was irrelevant to most of the users. The only important point was to have every module visible without scrolling. The general arrangement of the modules was built by using the tabs of the portal, which were given task oriented names.
- **Used portals.** The users equally chose iGoogle and Netvibes. One user criticized the slow processing of Netvibes although he was satisfied in general.

- **Assets and drawbacks of portal modules.** A great benefit of our approach is the high flexibility in structuring an own workspace. It is easily possible to combine important functions and to monitor interesting objects. Furthermore a visual cross linking between two or more workspaces becomes feasible without opening several browser windows.

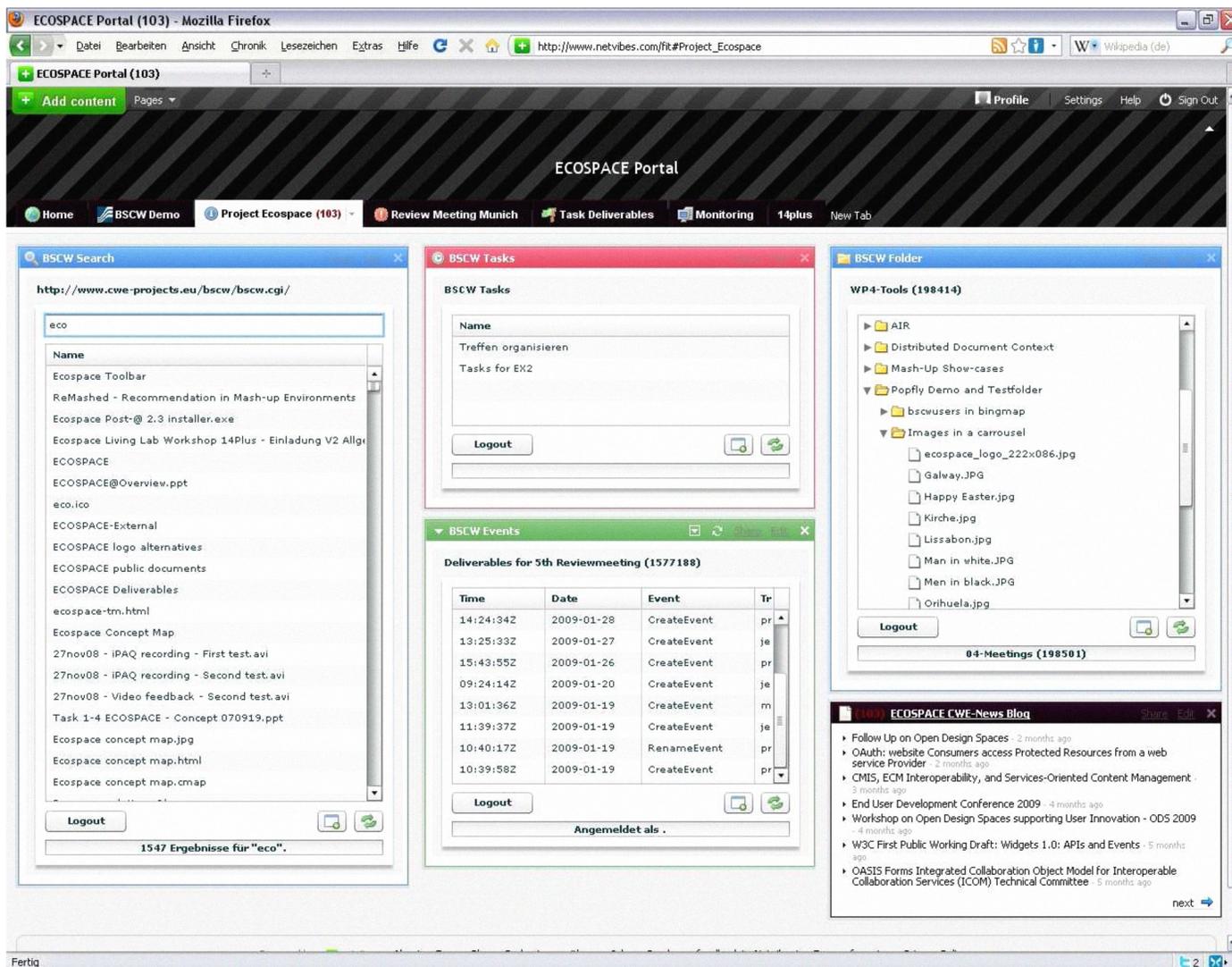
But there are some minor disadvantage, like the limited functions and space of the modules. Also the long loading time was often mentioned. A major point is the change in the operation method.

Application Scenarios

The integrated user interface of BSCW provides one view for all users, whereas the portal approach provides an individual view for every single user. This possibility of heaving different views leads to the question what views are appropriate. During the evaluation, the users report five different scenarios for their portal settings:

Figure 7. Screenshot – Meeting portal view

- **Project Portal** includes Search, Events, Folder, Tasks and RSS modules. Every module has a specific context to the project. A folder for example does not show a workspace of the user but a workspace of the project (see figure 6).
- **Meeting Portal** includes Folder, RSS, Search, Weather and Map modules. This view is set up for a meeting with a folder in a deeper project hierarchy. In addition to the BSCW modules, third party modules are combined with them like weather and map modules (see figure 7).
- **Task View** includes Events, Folder, RSS and Upload module, just to fulfill a specific task.
- **Monitoring Dashboard** includes several Event and RSS modules and a Search module. Thereby the user stays informed about current happening events.
- **Integrated View** includes several folder modules and a RSS module. Whereas the folder modules include view from different projects that might also be on different



servers.

This list indicates that the users have understood the concept that they can arrange the information and services of a cooperative environment within a new context, thus being able to individualize the access and use of a cooperation environment to their specific needs. Furthermore the approach presented in this paper goes beyond a folder/document based access by enabling the user to arrange the working environment as a mixture of information and services from different internal BSCW-sources as well as external sources (Fig 7) within a single portal. We believe that this provides a new concept for the organization of a collaborative working environment [10].

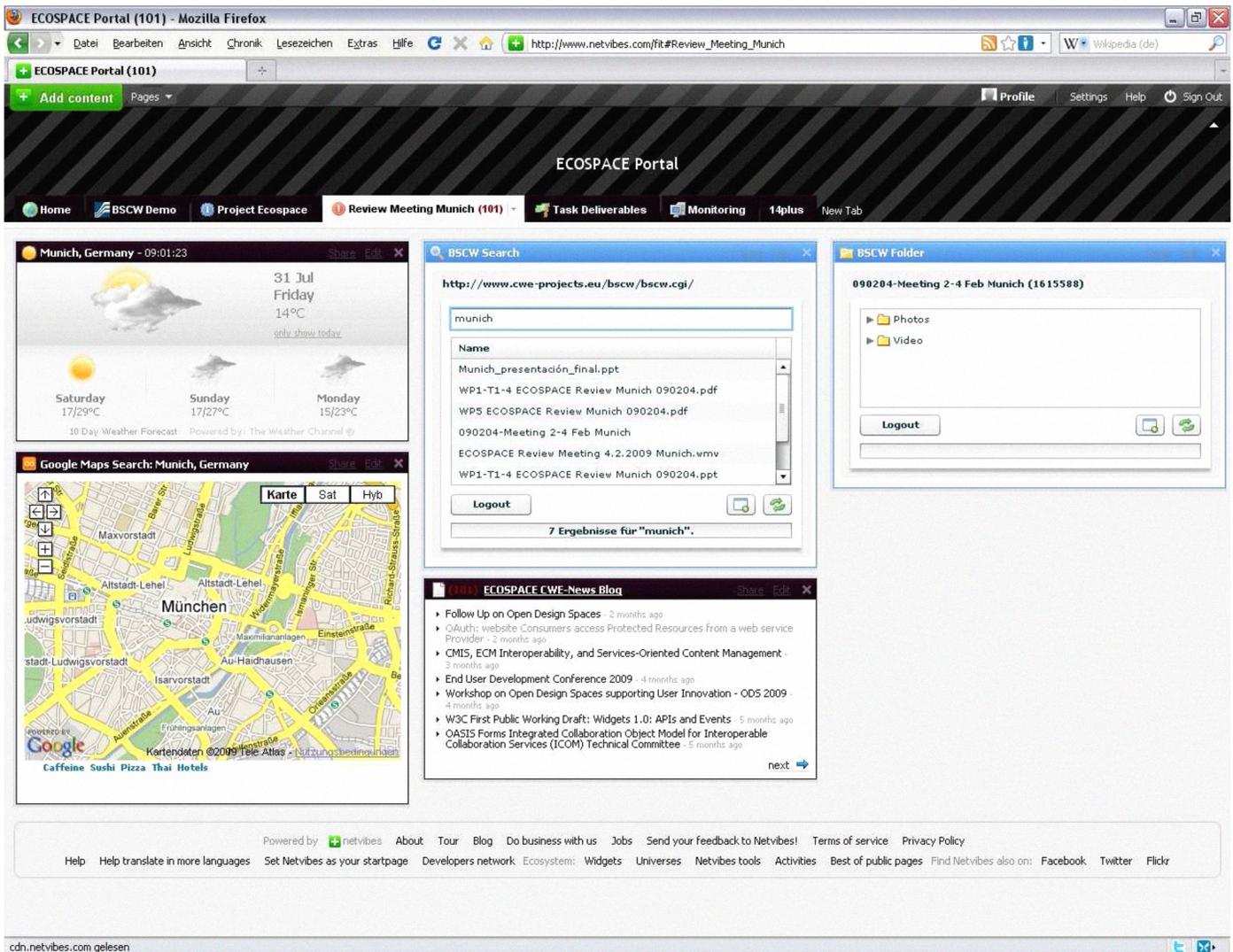
VIII. CONCLUSION

In this paper we have introduced our approach of portal modules. Like every new system, our modules were first unfamiliar to the users, and they preferred foremost their familiar methods. Nevertheless, the evaluation yielded positive results as well as feedback that lead to improvements during the DIA-cycles and continuing development. Judging from our results, we expect a growing demand for BSCW modules along a rising popularity of personal portals like iGoogle and Netvibes.

ACKNOWLEDGEMENTS

The work presented in this paper has been partly funded by the European Commission as part of the ECOSPACE Integrated Project.

Figure 8. Screenshot – Project portal view



REFERENCES

- [1] Abdelnur, A., Heppner, S. Portlet Specification Version 1.0 (2003), Sun Microsystems & IBM Corporation.
- [2] Appelt, W. What Groupware Functionality do Users Really Use? In: Proceedings of the 9th Euromicro Workshop on PDP 2001, February 7-9, 2001. Mantua: IEEE Computer Society (2001), Los Alamitos.
- [3] Cáceres, M. Widgets 1.0 – W3C Working Draft (2009). <http://www.w3.org/TR/widgets/>.
- [4] Dourish, P. The Appropriation of Interactive Technologies - Some Lessons from Placeless Documents. Computer Supported Cooperative Work, Volume 12, Issue 4 (2003), 465 – 490.
- [5] Koch, M. & Teege, G. Support for tailoring CSCW systems: adaptation by composition. In: Proceedings of 7th Euromic workshop on parallel and distributed processing, Funchal, Portugal, IEEE Press (1999), 146-152.
- [6] Mackay, W.E. Users and Customizable Software: A Co-Adaptive Phenomenon, Ph.D. Thesis, MIT, Boston (MA) (1990).
- [7] Malone, T.W., Lai, K.-Y. & Fry, C. Experiments with oval: A radically tailorable tool for cooperative work. In: Proceedings of CSCW. Toronto, Canada, ACM Press (1992), 289–297.
- [8] Negroponte, N. Being Digital. Alfred A. Knopf, New York (1995)
- [9] Norman, D.A. Emotional Design. Basic Books, New York (2004)
- [10] Prinz, W., et al. ECOSPACE – Towards an Integrated Collaboration Space for eProfessionals. in CollaborateCom 2006. Atlanta: IEEE press.
- [11] Roseman, M. and S. Greenberg (1996). TeamRooms: Network Places for Collaboration. Conference on Computer Supported Cooperative Work (CSCW'96), M. S. Ackermann, ACM Press, pp. 325-333. (2004)
- [12] Wulf, V., P. V., and M. Won, Component-based tailorability: Enabling highly flexible software applications. Int. J. Hum.-Comput. Stud., 2008. 66(1): p. 1-22.