Abstract

Community support platforms are gaining more and more interest in areas ranging from leisure support and customer support in electronic commerce to knowledge management in enterprises. However, current solutions usually are built as proprietary systems or as add-ons to systems designed for other purposes. The platforms are usually not very customisable and interoperable and are not utilizing the full potential of the community support idea. We have introduced community platforms in several domains and have derived some requirements in these projects. These requirements are mainly related to different aspects of usability. In this paper we present the requirements and motivate an architecture for community support systems designed to fulfil these requirements.

1 Introduction

Several types of applications currently advertise their new “community support” features. Under this label Web platforms and electronic commerce systems are equipped with annotation functionality and various communication features.

In general, community support implies all methods for supporting communication in a group of people. It includes support for direct communication, support for indirect information exchange and support for matchmaking. Community support is closely related to the application areas of knowledge management, customer relationship management and change management since these also deal with supporting communication in a loosely coupled group of people.

The use of networked computers to support communities can be traced back to the beginnings of the Internet: The second service in the initial Internet, the file transfer service was soon “misused” to transfer messages from one person to another – email was invented (Hafner & Lyon 1996). Quickly mailing lists followed and Newsgroup services were created – both on the Internet (Arpanet) and on alternative networks formed of loosely connected computers (e.g. FidoNet).

But community support is not limited to computers. Support for the building and the maintaining of communities can be classified in classical approaches like private letters, leaflets, magazines, paper whiteboards, specialized radio and TV programs, and approaches based on networked computers (bulletin board systems, MUDs, MOOs, “community networks”).

Both support types, the classical and the electronic ones, have their advantages and disadvantages. For classic media the advantages are availability, familiarity, and ease of use. For electronic media the advantages are dynamicity, speed, ease of replication, and distribution; disadvantages are barriers to usage, problems with access, and lack of availability.

We at Technische Universität München are working on computer-based community support for several years now. We have implemented and are operating several community support platforms. In these projects we have identified requirements for community support and have developed a community support system architecture which takes these requirements into account (Koch and Lacher 2000; Koch and Wörndl 2001; Koch 2002a).

In this paper we present the basic requirements for community support systems (Section 2) and introduce the architecture for community support systems, which was developed from these requirements (Section 3). The architecture mainly addresses modularization and integration. For modularization we present a set of core functionalities and concentrate on data models and services for user profile and content management.

When talking about community support we distinguish (community) support platforms and (community) support systems. While a support system is a piece of software that can be used to support communities, a support platform is an instance of such a software (together with specialized hardware potentially needed for interaction). One often also finds the terms ‘community software’ for community support
systems and ‘online community’ for (Web-enabled) community platforms.

2 Requirements for community support software

In general a community is a group of people who share some interest or belong to a common context. In addition to this characterization of a community as a describing identity for a set of people one usually also requests an ongoing rhythm of social interaction (Mynatt et al. 1997) and collaborative action.

Early sociological work points out that communities always need a locality and interaction (Hillery 1955). While the demand for a common physical locality is no longer seen necessary, the demand for interaction is still valid. However, no active interaction among all community members is required but rather the possibility of interacting with the rest of the community (the existence of a communication medium). In more practical terms this possibility of interaction implies the existence of a common communication channel, of common protocols and awareness of the existence and of the membership in the community.

Another characterization is that communities are based on the will to exchange knowledge. Ishida (1998) summarizes this in the following quote: “In a community, people want to know what the others know.” This issue can be extended to the request for collaboration in a community. A community is not just a set of people who have something in common and who have the possibility to communicate, but of people who are willing to help each other, who are collaborating to the advantage of all.

To summarize so far, a community is characterized by:

- a boundary (common interest, common idea, common context),
- a sense of membership,
- ongoing interaction, and
- collaboration, mutual support.

In addition to the collaboration itself the main activities in communities are communication and finding people to communicate with. Hence, community support can be seen as “communication and matchmaking support”. Community support systems have to provide a “medium” that can be used for the interaction among the members. This can be a clubroom with bulletin boards, a club magazine, regular physical meetings or a solution based on networked computers.

For deriving requirements for community support systems we started looking at the main activities in communities and at general characteristics of communities.

Community support is communication and matchmaking (awareness) support.

The main activities in communities are finding other people and communicating with other people in order to collaborate or help each other. In this context, the term communication is understood in a very broad sense. Communication can be classified as direct (email, SMS, chat, etc.) or indirect (publishing information for potential prospects, retrieving previously published information from the community information space etc.), synchronous or asynchronous, automatically triggered or manually triggered.

In summary, the following basic support concepts can be derived:

- Providing a medium or channel for direct communication and for indirect exchange of information objects or comments on objects within the common scope (the information space) of the community. The information channel can be enhanced with features that use information about the community member to do (semi-)automatic filtering and personalization.
- Providing awareness of other members and helping to discover relationships (e.g. by visualizing them). This can help to find possible cooperation partners for direct interaction (matchmaking, expert finding).

Community support platforms have to provide such functionalities for their members.

People usually are members in different communities.

For community support platforms this means using different platforms in parallel should be easy. One problem with all current community support solutions is that they do not offer standard interfaces for interaction and that data cannot be exchanged or be used by more than one application. Users have to enter their profile information again and again and have to interact explicitly with the different applications using different user interfaces. Even with intuitive Web user interfaces and a single-login and profile exchange feature the effort of visiting different platforms is often too high for a user. So there is an additional need for non-interactive interfaces to easily publish to different platforms and easily collect information and notifications from different platforms. We propose to allow access of information agents to the platforms to solve this issue.

Community interaction happens not only when people are sitting in front of their desktop PCs.
Access to community platforms has to be available from everywhere and during any task. So mobile and ubiquitous access to the platforms is needed. Core services and user interface modules have to be separated and new user interfaces have to be introduced. This includes user interfaces that can be used by several people at once (like wallpaper-like sensitive large-screen displays). Electronic community support has been, till now, determined by boundaries of stationary computers and desktop based user interfaces. Ubiquitous computing, i.e. new user interfaces and the disappearing computer, and mobile computing are addressing these boundaries and offer possibilities for enlarging the reach of community sup-port systems. In addition to enlarging the reach, mobile interfaces open completely new fields for community support – new functionalities and new scenarios can be contrived. Some projects have already started to tackle this objective. For example the project Campiello (Agostini et al. 2000, Grasso et al. 2000, Koch et al. 1999) was targeted towards the development of a community support system for the tourist domain with interfaces in the tourist towns.

Communities can be created ad-hoc and by anybody. Therefore, it has to be possible to easily set up and customize a support platform for new communities. A modular and customizable architecture is needed to make this possible.

Personal interaction and the knowledge about each other is very important in a community.

Community support systems have to provide personalized access and access to other users profiles. For both tasks user profile information is needed. So community support systems have to deal with profile acquisition and profile storage. As with other personalization systems the main issues in this area are the acquisition of user profiles (“cold-start problem”) and the definition of access rights for ensuring privacy (Schubert & Koch 2002). However, the need for making user profile information readable for other users adds new demands on the representation of the user profile information.

### 3 Community Support System Architecture

In addition to providing the needed support functionalities a community support system has to be highly configurable to make the instantiation of different platforms possible. It has to be possible to exchange or reuse information (especially user profile information) among platforms based on the same software and with other systems, and the system should be accessible by different kinds of user interfaces and through information agents.

In the Cobricks project we have designed and built a support system for communities, which can be used stand-alone or be integrated into existing portal or community support frameworks.

#### 3.1 Modularization

The key issue in the design was to identify generic service modules with open interfaces and to implement them to be highly configurable. From the general community support characteristics and from our experiences in the different projects we have derived the following core services for this architecture:

- item management (content management)
- annotations management
- user profile management
- community management
- personalization and filtering
- matchmaking
- messaging, event management

**Item management** is about managing contributions from members and from community moderators. Items are typed content objects with meta-information based on their type. The meta-information is used for searching and for filtering (personalization). **Annotation management** is about managing member annotations to items (ratings or comments). **User profile management** deals with managing information about users. While user profile management handles single profiles **community management** is about managing the member list(s). User profiles and item meta-information are combined in the **personalization modules**. In addition to generating recommendations on demand these also do proactive identification of interesting information and generation of notifications (e.g. newsletters). **Matchmaking services** compare user profiles to identify like-minded users. Finally, the **messaging service** handles messages sent from services or users to members of the community and deals with availability management and filtering.

The modules are implemented independent from user interface issues and can be used from a generic web portal module, from modules managing new user interfaces or from existing portal and community frameworks. To make this possible the modules return results in XML, which then is further processed by the calling modules. In addition to integrating the modules in existing user interfaces or creating a new Web user interface for them we also allow user information agents to directly interact with the modules. Therefore, the modules provide a very high-level (agent-like) interface.

---

\(^1\) Cobricks = (Software) bricks for supporting communities. See www11.in.tum.de/proj/cobricks/ for more information.
3.2 User Profile Management

User profiles are used to do personalization, to do matchmaking and to present a user to other users. In contrast to the current state of art the profile therefore has to be represented in a generic way. The profile also has to include different kinds of (usually complex) information ranging from user names to personal information manager (PIM) information like date book information. The following types of information about a user have been identified to be relevant for providing community support:

- demographic information
- interest information
- ratings
- browsing and shopping (transaction) history
- relationship information
- preferences
- PIM information (e.g. date book)

For the generic representation of user profiles we are currently defining and refining an XML-based profile structure with some controlled vocabulary, which is connected to item ontologies and the community platforms category management.

The user profiles are acquired through directly requesting information from the users (in fill-in-forms or in dialogs) or by watching the users behavior. However, the acquisition itself does not solve the “cold-start-problem”. This means that users expect good recommendations from the beginning but the system is only able to provide recommendations after having spent some time asking or watching the user. Ideas to solve the cold-start problem are importing user profiles from other sources or iteratively collecting information.

To support the importing of user profiles we are working on a solution that allows different platforms to share the same user profiles. We have proposed a user profile management architecture that separates the profile storage from the community services (Koch and Wörndl 2001, Koch 2002b)

Separating service and profile raises some issues concerning privacy and access rights, which are addressed together with the privacy issues coming from providing user profile information to other users.

3.3 Item Management

All modules in the community support platform need to be highly customizable and easy to extend. This is especially true for the item management module. It should be easy to introduce and use new item types without changing the software. Therefore, high-level interfaces and open data structures are needed (same as for user modeling).

The most important issue in this is the design of a generic information representation model. In our architecture we here have identified the following information objects:

- item
- (item) annotation
- category
- person
- event / message

Items are typed content objects with meta-information based on their type. They are represented in XML following an easily extensible ontology. This concept of representing data objects in XML based on ontologies and controlled vocabulary is also used for the other information objects.

For the interfaces of the basic modules we have introduced a generic query mechanism for information objects based on the data model mentioned before. The queries currently are defined as XPath terms. For future versions we are also looking into F-Logic and other standard query languages.

4 Prototype Platforms and User Interfaces

The requirements that were briefly discussed in Section 2 and the architecture discussed in Section 3 have been derived from and been implemented in several application domains. The two main domains are universities and leisure support.

In the university domain we are working on communication and announcement platforms for different groups and on knowledge management platforms for teams (Koch et al. 2001a; Koch et al., 2001b). The largest platform we are operating is a Web community for the computer science faculty at our university with several thousand users. Main insights from the university domain were that there is a large need for interoperability – especially integration in existing infrastructures, but also for adaptability and extensibility.

For leisure support we started with recommender platforms in the tourist domain (Project Campiello) and continued with mobile lifestyle support (Project Cosmos). The main requirements learned here were easy access, easy tailoring and openness to different access media (including mobile devices and large screen displays).

The issue of ubiquitous access to community support systems is tackled in the leisure support area. In the Campiello project we have used paper-based input devices and large screens for displaying information to the community (Koch et al. 1999, Agostini et al. 2000, Grasso et al. 2000). In the Cosmos project we are working on mobile community support – i.e. using mobile devices (PDAs with mobile network access and mobile phones) to interface with the community.
support platform. In this scenario we are especially interested in location based communication and matchmaking (awareness) support (Hillebrand et al. 2002, Koch et al. 2002, Reichwald et al.).

5 Conclusions

In this paper we have briefly outlined some requirements for community support systems and have presented the basic design ideas for a community support system architecture developed at Technische Universität München. Major suggestions for the platform design were drawn from reviewing the basic characteristics of the application area, communities and community interaction.

The main issue with the architecture was to build a flexible and highly customizable solution for setting up community support platforms. This was addressed by a clear modularization of services and by introducing a generic information representation for items, item annotations and user profiles. A user profile exchange allows for the provision of adequate member representations and for easy moving between platforms. The solution is in use with a local portal framework and with the Cassiopeia Community Application Server.

For the future we are planning to continue the work on data structures, extend the personalization functionality, and concentrate on new user interfaces for community support.

References


