DISTRIBUTED DEVELOPMENT OF A DISTRIBUTED VIRTUAL COMMUNITY SYSTEM

Tom Gross[1,2], Wido Wirsam[1]

The CYCLADES open collaborative virtual archive environment is a virtual community system that supports knowledge workers in searching and managing e-print documents from a large number of digital libraries. Users can share their results in existing communities as well as find new collaborators with similar interests. Four different teams with different scientific backgrounds developed the software. The resulting system runs distributed on four different hosts. The communication among the software pieces as well as the user interface is based on standard Internet technology. In this paper we will present the design, specification, and implementation of this virtual community system.

1 Introduction

Scientific Research nowadays tends more and more to be interdisciplinary and immediate. Collaboration among scientists, however, is strongly restricted by geographical, organisational and subject area boundaries. Occasions where people working on similar subjects can efficiently exchange and distribute their findings are rare. Usually a few times per year conferences take place, where this intense communication takes place. The CYCLADES system supports knowledge workers with a technological approach that helps to intensify their collaboration in their everyday work. CYCLADES provides an open collaborative e-print archive environment to support communities of knowledge workers online. Electronic pre-print (e-print) archives are important vehicles for the dissemination of preliminary results and non-peer-reviewed grey literature. There is a growing need for easy retrieval of information from diverse sources. CYCLADES addresses these issues and supports the transition of e-print systems into the genuine building blocks of a transformed scholarly communication model by developing a set of leading-edge technologies. The archives that are connected to the CYCLADES system must be compliant to the OAi specifications. The OAi consists of a technical and organisational framework designed to facilitate the discovery of content stored in distributed e-print archives [5]. Basically, the OAi metadata harvesting protocol supports the communication between

[1] Fraunhofer Institute for Applied Information Technology, 53754 St. Augustin, Germany
{tom.gross, wido.wirsam}@fit.fraunhofer.de

[2] During the time of the writing of this paper, the author was visiting professor of media informatics at the RWTH Aachen, Ahornstr. 55, 52056 Aachen, Germany

different digital library components via a simple, yet powerful protocol. Any archive that fulfils these interface specifications can be integrated into the system by any user. So the knowledge base of CYCLADES is designed to grow continuously.

In this paper we will reflect on the process of building the CYCLADES environment. We will particularly present the approaches for designing the functionality of the overall system as well as of the individual services; we will describe the form of the formal specifications we chose and give some example specifications; and we will elaborate on the implementation of the environment.

2 Design

In this section we will describe the basic functionality of the CYCLADES system as well as the basic decomposition of the overall system into individual services. More details on the functionality of the overall CYCLADES system in general and the support for virtual communities in particular, as well as an overview of the different services can be found in [3].

2.1 Functionality

The CYCLADES system offers a broad range of functionality for both individual scholars, who want to search and browse in open archives as well as for communities of scholars who want to share and exchange search results. This functionality was designed by a multi-disciplinary and distributed team with backgrounds in digital libraries, databases, information retrieval, Web-based systems, as well as computer-supported cooperative work and virtual communities. After a first suggestion for the basic functionality of the system was finished, this functionality was presented to prospective users, and their feedback was captured via a Web-based questionnaire. Results from this questionnaire were reported in Gross and Kreifelts [4].

The CYCLADES system integrates a set of functionalities that support the user when accessing very large virtual e-print archives with: functionality for efficient and effective retrieval of relevant information from many large, distributed and multi-disciplinary digital archives; feedback on the degree of relevance of the retrieved information; regular information about new publications in the archive environment which are relevant to the users' interests; automatic retrieval of users' long-term information needs (user profiling); as well as rapid dissemination of the search results world-wide. A special set of features provides communities of scholars with functionality for: the dissemination of relevant information to community members in the form of recommendations, which are based on collective profiles and behaviour; very quick on-line annotations on research results published by members of the community; carrying out community services such as peer review, which requires the annotation of on-line manuscripts by reviewers, and the sharing of these annotations among editors, authors, and others; as well as enabling community members to learn from, contribute to, and collectively build upon the community's knowledge.

2.2 Services

The archive environment is composed of a large number of heterogeneous, multidisciplinary archives and supports interoperability between them. It is based on the Open Archive initiative (OAi) [5]. One component of CYCLADES—the Access Service—builds the bridge to the archive environment and makes a huge amount of distributed knowledge stored in many digital libraries accessible to the CYCLADES users.

The different services communicate over an interface that is based on standardised Internet communication technology (XML-RPC). This architecture allows the different servers to be optimised to achieve best performance in the specific task for which it is responsible. Highly time-consuming operations (e.g., re-indexation of archives) can be performed in the background without decreasing the overall system performance. This specification also gives excellent flexibility in the adaptation, updating or optimisation of services or the integration of a new functionality. The CYCLADES environment consists of the following standard services:

- The *Mediator Service* (MS) plays a central role among the services. Every other service that is integrated in the system registers with the MS. Whenever a service communicates to another one the MS is asked for the appropriate service and address. This gives a great flexibility and transparency to the distributed system. The MS also manages the registration and login of users to the CYCLADES system.

- The service responsible for accessing all connected archives is—as mentioned above—the *Access Service* (AS). It harvests, indexes and retrieves metadata records from the underlying archives.

- The *Collaborative Work Service* (CWS) provides a folder-based environment for managing metadata records, queries, collections, external documents, and annotations. The CWS supports users with functionality for the management of folders and their contents and for cooperation among community members, including special awareness features reflecting the groups' activities in the shared information space.

- The *Rating Management Service* (RMS) handles ratings of items stored in the CWS and provides those information to the users as well as via a query interface to other services of CYCLADES.

- The *Search and Browse Service* (SBS) supports searching for records from the various collections, formulating search queries, and browsing the resulting metadata records.

- The *Filtering and Recommendation Service* (FRS) provides personalised filtering of queries and query results, provides recommendations of relevant records, collections, users and communities. The FRS analyses a user's behaviour and data, generates a user profile describing the user's interests, and provides information according to this profile.

- The *Collection Service* (CS) supports the creating and editing of collections. Collections partition the information space according to the users' interests, and make the individual archives transparent to the user. The CS allows each user to create and maintain a personal list of collections consisting of a sub-set of all the collections available.

3 Formal Specifications

The formal specification of the overall CYCLADES system was done with a light-weight variation of a UML approach [1, 2]. That is, several modelling techniques such as use cases, interaction diagrams, and class specifications were used, but the syntax was simplified. This simplification was possible, because of several face-to-face meetings of the distributed design and implementation team and was chosen in order to reduce the effort for the specification. We will present some examples of use cases, interaction diagrams, and class specifications below.

3.1 Use Cases

A use case can be defined as: 'a set of scenarios tied together by a common user goal', while a scenario is: 'a sequence of steps describing an interaction between a user and a system' [2, p. 39f]. Since the CYCLADES systems offers a broad range of functionality for single-users and for communities, a considerable number of use cased had to be developed. Therefore, we decided not to use graphical representations of use cases. Rather verbal descriptions of the use cases were written and, in some cases, combined with activity diagrams.

The use cases were grouped into use cases for the single user, use cases for a community, use cases for a project, as well as some special use cases related to management activities such as use cases for collection management, and use cases for archive management. An example use cases for the single user is joining the CYCLADES environment. Figure 1 shows this use case.

Register as new user (the user gives her email address and in turn gets a user name and password with which she can login)

Login (the user enters her user name and password. After login, she is presented with her home folder)

View and edit user information

Figure 1. Joining the CYCLADES environment use case.

For the first step—Register as new user—an activity diagram was drawn; the latter two steps were only described verbally. Figure 2 shows the corresponding activity diagram, which shows further details for the first stop.

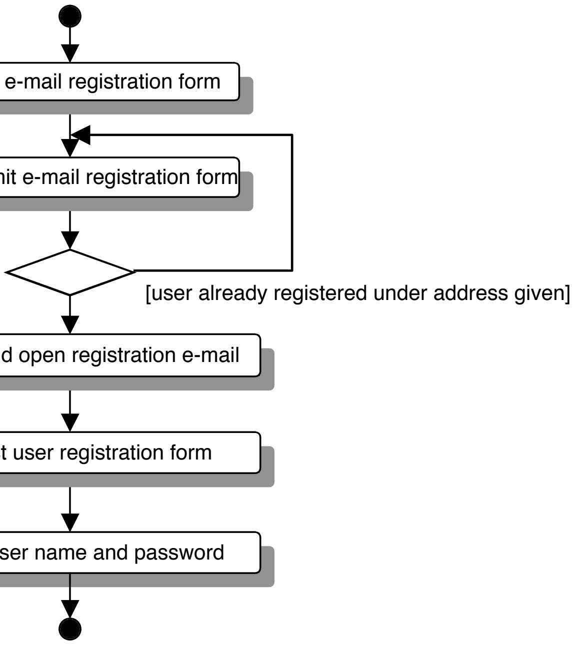
Request e-mail registration form
(Re-)Submit e-mail registration form
[user already registered under address given]
Receive and open registration e-mail
Request user registration form
Submit user name and password

Figure 2. Register as new user activity diagram.

An example use cases for a community is setting up a new community, including the management of its members. Figure 3 shows this use case.

Create a new community folder (in her home folder, given the appropriate rights, the user can create a new community by creating a community folder; she enters a description for the new community and defines if other users can subscribe without or only with invitations; as creator of a community, she automatically gets the rights to administrate the community and thus becomes an administrator of the community)

Invite and kick out members of the community (the administrator of a community can invite other users, kick out other users, and define access rights for members)

Figure 3. Setting up a new community use case.

For this use case and its two steps no activity diagrams were produced, but further verbal specifications were done. Figure 4 shows an example of the step invite new members.

Invite new members

The administrator hits the 'Add member' function in the home folder of the community. She is presented with a form where to specify the invitees and the message with which the invitees are invited.

There are two alternatives for specifying a new member to invite, either by user name for already registered users or by email address for a future member that still has to register. The invitation of an existing user has the effect that the community home folder appears as a subfolder of the invited user's home folder. The invitation of a new user results in a similar procedure as if this person had tried to register herself (cf. use case CO 3-1 'Register after invitation'). There are also two alternatives for the message to be sent to the invitees: it may be an automated message for which the administrator has to select the appropriate language, or it may be a specific message to be specified by the administrator herself. In the latter case she may also have this message only be sent to the invitees who are no registered users or to all invitees.

By submitting this form, all invitees become members of the community. Non-registered invitees will receive an invitation to register. Invitees who are already registered users will have the community home folder with all its subfolders in their personal home folder.

Figure 4. Invite new members step.

3.2 Interaction Diagrams

Interaction diagrams are: 'models that describe how groups of objects collaborate in some behaviour'; they typically: 'capture the behaviour of a single use case' and 'show a number of example objects and the messages that are passed between these objects within the use case' [2, p.□67]. As the CYCLADES services have been developed at different sites in different countries, we decided to use interaction diagrams mainly for modelling the communication that occurs in a use case, in which different services are involved. Basically interaction diagrams can be drawn as sequence diagrams or as collaboration diagrams; in sequence diagrams: 'an object is shown as a box at the top of a dashed vertical line', and in collaboration diagrams: 'the example objects are shown as icons' [2, p.□67ff].

As an example we show the interaction diagrams for the user registration. The registration involves two CYCLADES services—the mediator service, which provides the user interface and database for the central registration, and the collaborative work service, which upon registration creates a user object and a home folder for the respective user. In order to simply the interaction diagrams we created two diagrams for the registration—one for the registration from the mediator service's perspective (cf. Figure 5), and one for the registration from the collaborative work service's perspective (cf. Figure 6).

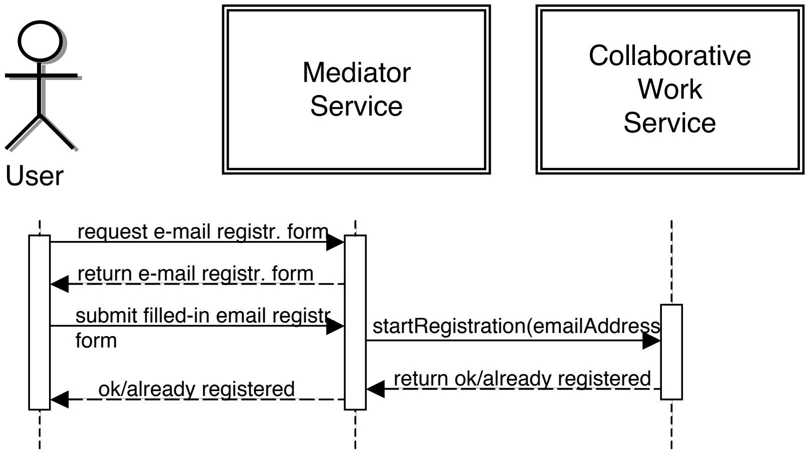
User
Mediator Service
Collaborative Work Service
request e-mail registr. form
return e-mail registr. form
submit filled-in email registr. form
startRegistration(emailAddress
return ok/already registered
ok/already registered

Figure 5. Registration interaction diagram (MS perspective).

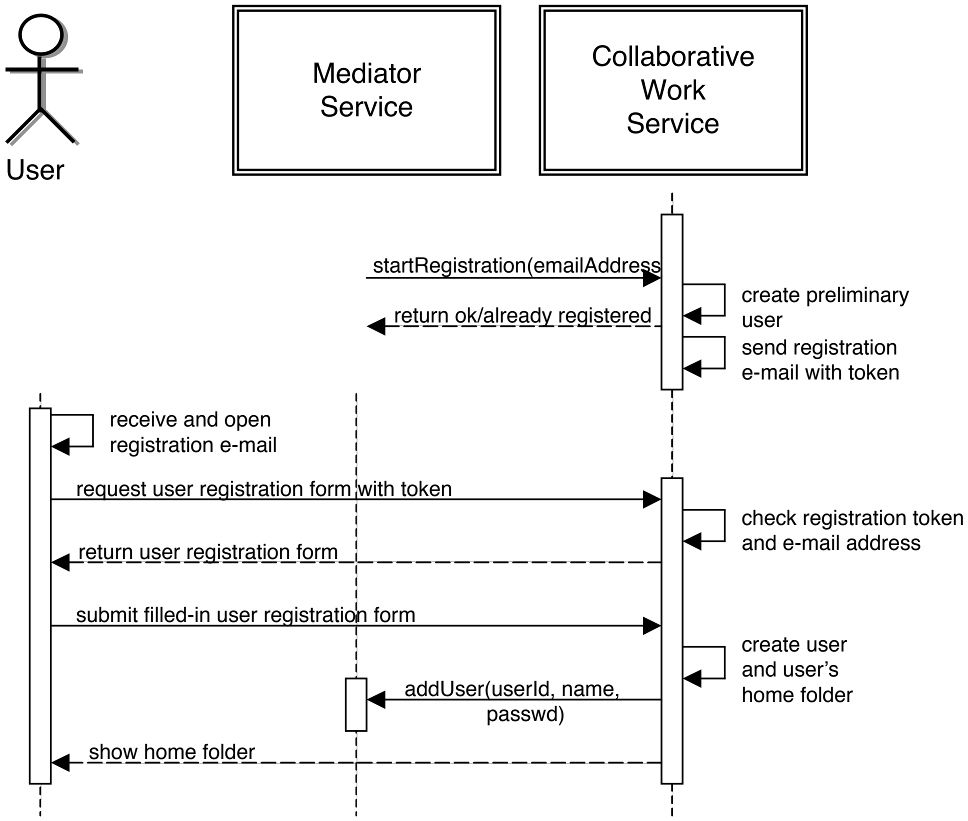
User
Mediator Service
Collaborative Work Service
startRegistration(emailAddress
return ok/already registered
create preliminary user
send registration e-mail with token
receive and open registration e-mail
request user registration form with token
check registration token and e-mail address
return user registration form
submit filled-in user registration form
create user and user's home folder
addUser(userId, name, passwd)
show home folder

Figure 6. Registration interaction diagram (CWS perspective).

As a further example we show the interaction diagram for the invitation of a user to a community or project folder. The invitation again involves the mediator service, and the collaborative work service. This interaction diagram is particularly interesting, because it shows a cooperative process with two users involved—the inviter, and the invitee. Figure 7 shows the corresponding interaction diagram.

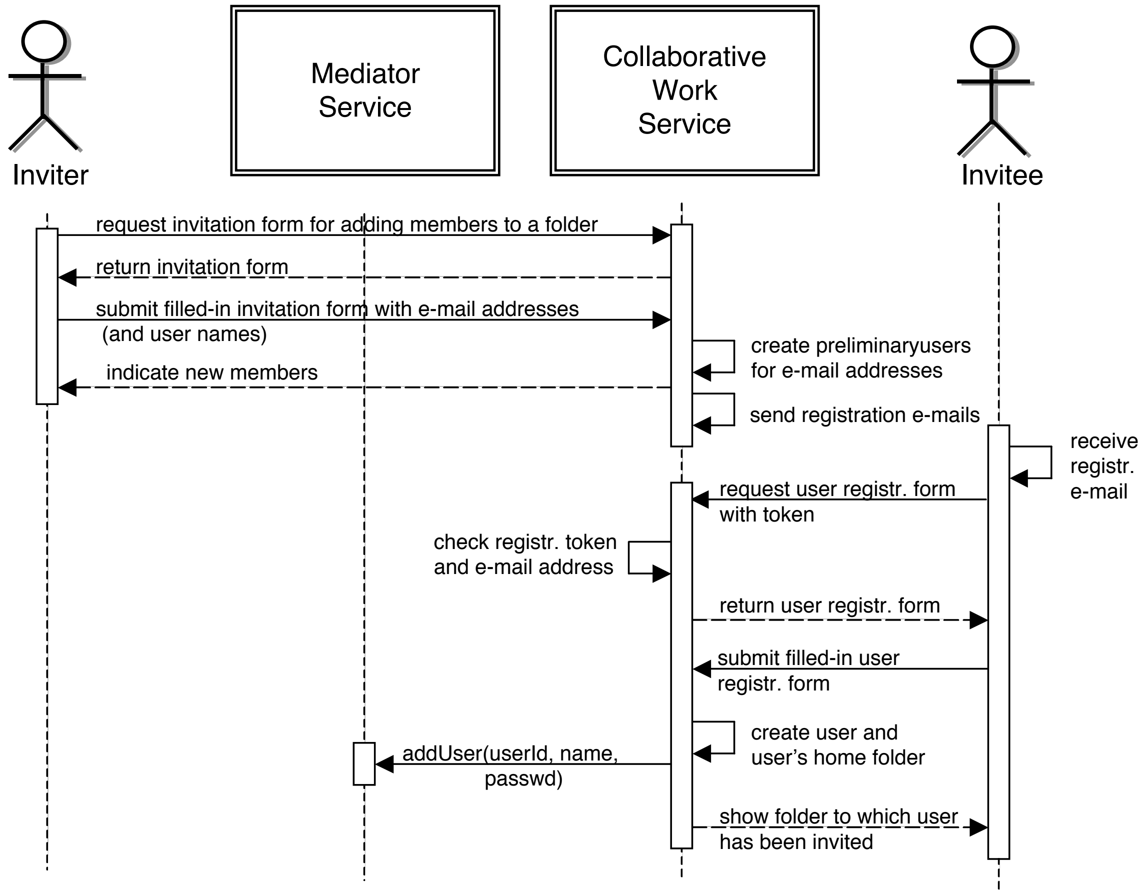
Mediator Service
Collaborative Work Service
Inviter
Invitee
request invitation form for adding members to a folder
return invitation form
submit filled-in invitation form with e-mail addresses (and user names)
create preliminaryusers for e-mail addresses
indicate new members
send registration e-mails
receive registr. e-mail
request user registr. form with token
check registr. token and e-mail address
return user registr. form
submit filled-in user registr. form
create user and user's home folder
addUser(userId, name, passwd)
show folder to which user has been invited

Figure 7. Invitation interaction diagram.

3.3 Class Specifications

The specification of the classes had to be very flexible, since the different partners were using different programming languages and environments. We, therefore, decided to have simple verbal descriptions of the signatures of the classes of the individual services. Figure 8 shows the beginning of the description of the signatures of the collaborative work service.

4 Implementation

The CYCLADES system consists of several autonomous yet interoperable software pieces called services. The services run on different computer systems in different locations. The Mediator Service (MS) plays a central role for all services. Any participating service has to register at the MS to be able to communicate with any other service. For all services to cooperate, the interface had to be well defined. The interfaces of the different services were implemented as an API that can be accessed by all other services. The method names and the parameters were encoded using XML-RPC. This allows providing all functionality to any other service independent of the underlying programming language.

```
CollaborativeWorkService
id
   Description: this is the unique ID of the service.
collectionNames
   Description: this is a list of pairs containing the ID and name of
   the currently existing collections.
boolean startRegistration(emailAddress)
   Description: this method may be invoked in order to start
   the registration process for a user with a given email address.
   Input:  emailAddress:  an email address.
   Output: true, if no user has yet been registered under
           the email address given, false otherwise.
folderId* getFolders(userId)
   Description: this method may be invoked in order to get a
                list of IDs of folders to which a specific user has access. \\
   Input:  userId: user ID.
   Output: a list of folder IDs.
name getName(folderId)
   Description: this method may be invoked in order to get
                the name of a folder.
   Input:  folderId:  a folder ID.
   Output: a string containing the folder name.
…
```

Figure 8. Signatures of the collaborative work service (excerpt).

For instance, the Collaborative Work Service (CWS) notifies the Filtering and Recommendation Service (FRS) when community folders are created or deleted, or when recommendation preferences are changed via the FRS API methods `addfolder()`, `deleteFolder()`, `setRecommendationPreferences()`.

Figure 9 shows as an example the remote procedure call of the method `saveRecommendedRecords()`. If this feature is enabled by the user or community the Filtering and Recommendation Service (FRS) regularly recommends items that match the users' interests using this method. The FRS calls this method that is provided by the CWS to store records in a folder. The transmitted XML call contains the encoded method name and two arguments: the folder ID indicating where the record has to be stored and an array of record identifiers.

```
<?xml version="1.0"?>
<methodCall>
   <methodName>service.saveRecommendedRecords</methodName>
   <params>
      <param>
         <value>CW665_12706</value>
      </param>
      <param>
         <value>
         <array>
            <value>
            AC143_oai_dc_oai:CompSciPreprints:Conferences/0202171
            </value>
         </array>
         </value>
      </param>
   </params>
</methodCall>
```

Figure 9. XML Remote Procedure Call.

The decision to divide the software into several independent services that fulfil a specific task has some obvious advantages, but it also requires some special treatment in the software development process. The advantages of a distributed software system are:

- Increase the overall performance because every service can use the available system resources in an optimal way.

- A grand flexibility of the systems functionality because the communication among the services is performed over a well specified interface. Any service can be exchanged by another implementation that is compatible to the interface specifications. New services may be integrated with little effort.

- The system environment of every service can be optimised to achieve the best results for the specific task. The different services of the CYCLADES system use different programming languages (Java and Python) as well as different Database technologies (mySQL, Berkeley DB and Oracle).

The following measures had to be taken in the development process of the distributed CYCLADES environment:

- The reasonable diversion of the software's functionality into pieces that can be operated mostly independent of each other and require only little well defined data exchange.

- A simple yet powerful communication protocol had to be found. We chose XML-RPC that engages remote procedure calls coded in XML and transmitted as HTTP requests.

- Because data redundancy can't be avoided the system has to define mechanisms to handle data inconsistencies.

- Different services provide their own user interfaces. To present the system as a whole to the user the user interfaces have to follow the same design guidelines.

5 Conclusions

Due to the open architecture of the system, it will be possible to implement and integrate new functionality into CYCLADES if this is required by the users. Third parties who want to use the system and may need to adopt it to match their individual expectations, can do so with little effort.

Because of the great flexibility and extensibility of the CYCLADES system, it is possible to modify or extend the system in various directions. Currently the system focuses on scientific researchers, that use digital archives to quickly gather information and discuss about it. But online e-print archives are not the only imaginable data source. Knowledge workers of any domain may use the CYCLADES system to search for information and exchange about their results.

At the time of the writing of this paper, the user evaluation of the CYCLADES system has just begun. A web based questionnaire was developed with the goal to find out whether the system matches the users' requirements. Questions about the overall usability and integration of the system are asked as well as questions about the different features and their realisation in particular.

Acknowledgements

The research presented here was carried out and financed by the IST-2000-25456 project CYCLADES, partly funded by the EC. We would like to thank our colleagues from the CYCLADES team at Fraunhofer FIT—and particularly Thomas Kreifelts—as well as at ERCIM, CNR, FORTH, and UNIDO.

References

1. Berners-Lee, T., Cailliau, R., Loutonen, A., Nielsen, H.F. and Secret, A. UML 2001: A Standardisation Odyssey. Communications of the ACM 29, 10 (Oct. 1999). pp. 29-37.

2. Fowler, M. and Scott, K. UML Distilled: A Brief Guide to the Standard Object Modelling Language. Addison-Wesley, Reading, MA, 2000.

3. Gross, T. CYCLADES: A Distributed System for Virtual Community Support Based on Open Archives. In Proceedings of the Eleventh Euromicro Conference on Parallel, Distributed, and Network-Based Processing - PDP 2003 (Feb. 5-7, Genova, Italy). IEEE Computer Society Press, Los Alamitos, CA, 2003. pp. 484-491.

4. Gross, T. and Kreifelts, T. Sharing and Exchanging Knowledge in Virtual Communities: The CYCLADES System. In Proceedings of the Tenth International Information Management Talks - IDIMT 2002 (Sept. 11-13, Zadov, Czech Republic). Universitaetsverlag Rudolf Trauner, Linz, 2002. pp. 71-81.

5. OAi. Open Archives Initiative. http://www.openarchives.org/, 2003. (Accessed 25/1/2003).