CYCLADES IST-2000-25456
An Open Collaborative Virtual Archive Environment

System Validation 5.1.1

Delivery Type: R
Number: 5.1.1
Contractual Date of Delivery: month 27
Actual Date of Delivery: May 19, 2003
Task: WP5

Names of Responsible: Gudrun Fischer, Sascha Kriewel, Saadia Malik

University of Duisburg-Essen, Duisburg
Department of Engineering Sciences
IIIS, Information Systems
D-47057 Duisburg Germany
E-Mail: fischer,kriewel,malik@is.informatik.uni-duisburg.de

Contributors

CNR-IEI:	Leonardo Candela, M. Elena Renda, Umberto Straccia
	Henri Avancini
University of Duisburg-Essen:	Gudrun Fischer, Sascha Kriewel, Saadia Malik
Fraunhofer FIT:	Tom Gross, Thomas Kreifelts, Wido Wirsam

Abstract: This report describes the validation of the CYCLADES prototype. Based on the tests specified in WP2, it details the functionality and performance testing of the system and the results.

Contents

Chapter 1

Introduction

This report is meant to document the validation of the CYCLADES system prototype. This system validation will be based on the list of efficiency and functionality tests that was previously defined in the CYCLADES deliverable 2.2.1, the Global System Architecture Report.

The CYCLADES system is an integrated environment that allows users a personalized and homogeneous access to distributed databases of document metadata that comply to the *Open Archives* protocol[1]. As such it is composed of a number of services, most of which a user can interact with through the use of a web accessible graphical user interface. The services of the CYCLADES system prototype to be tested are:

- Access Service
- Collaborative Work Service
- Collection Service
- Filtering and Recommendation Service
- Mediator Service
- Search and Browse Service

The validation of the CYCLADES system prototype was subdivided into two distinct tasks, that will be detailed in the remainder of this report:

- functionality verification
- efficiency evaluation

Firstly, the functionality of the system was tested. The aim of functionality testing was to verify that all the functionality requirements defined during the specification phase of WP2 have been met by the prototype implementation. More specifically, it was to be tested for the use cases defined in deliverable 2.2.1, that they can be fully completed by a user of the prototype. The tests of the system's functionality will be fully detailed in the second chapter of this report.

Secondly, the performance of the prototype implementaion was tested with the aim of determining the efficiency with which the system completes its tasks even under taxing circumstances, as e.g. heavy load, great amounts of data, or the loss of connection to one or more of its service components. For this purpose several experiments were conducted, using simulated user input under controlled circumstances, the results of which are detailed in chapter 3. The performance of the services

[1] http://www.openarchives.org

was tested at use case level, and where necessary and helpful for determining a possible loss in performance also at the method level.

These tests were carried out using a web browser for GUI based tests, or with special test clients for automatically generating HTTP or XML-RPC calls, that were used to simulate user input.

Where shortcomings in functionality or performance became visible, this was used as a basis for improving the current prototype to produce an overall more efficient and usable system.

Chapter 2

Functionality Validation

2.1 Introduction

The aim of the functionality tests as described in Chapter 4.6 of D2.1.1 is to verify if each of the use cases previously defined for the CYCLADES system can be successfully completed. If that cannot be done, at least the extend in which they can be completed will be determined and documented.

In the following section we briefly describe the types of uses cases we have, and give a list of all use cases that will be tested. A more detailed discussion of these use cases can be found in Chapter 2 of D2.1.1. Each use case was tested in the context of a specific service of CYCLADES. Usually this is the service where a use case is initiated if it involves more than one service. The remainder of this chapter is devoted to detailed descriptions of these functionality tests and their results.

2.1.1 The Use Cases

In the following we list the use cases that have been previously identified as actions that a user of the system should be able to perform. Each use case is listed with the service that it mostly concerns, and in the context of which it is going to be tested. We are using the following simple abbreviations for this:

AS: Archive Service

CS: Collection Service

CWS: Collaborative Work Service

FRS: Filtering and Recommandation Service

MS: Mediator Service

SBS: Search and Browse Service

2.1.1.1 Joining the Cyclades Community

To join the Cyclades Community it is necessary that a new user registers to the system. On registration the user receives a Cyclades account consisting of a user name, a password, and a home folder. With this pair of user name and password she can now login to the system. User information also has to be managed. Users can view and edit their details, or change their password.

- CWS: View and edit user details

- MS: Change password
- MS: Delete user
- MS: Login
- MS: Register as a new member

2.1.1.2 Folder Management

Within the home folder of a user, she can create, edit and destroy subfolder to store relevant records or queries. Each subfolder can have collections associated with it.

- CS: Select personal collection set
- CWS: Create folders
- CWS: Destroy folders
- CWS: Edit folder preferences
- CWS: Manage folders
- CWS: Update collections for folder

2.1.1.3 Gathering Records

Users can get new records for their folder in a number of different ways. Users explicitely search for records by issueing queries, or by personalized browsing that takes the profile of the current folder into account. These folder profiles are updated automatically by the system, and on user request. The CYCLADES system also recommends objects (i.e. records, collections, users, communities) to a user based on other users' ratings of records, and on the profile of a given folder.

- FRS: Recommend collections for folder
- FRS: Recommend records for folder
- FRS: Recommend users for folder
- FRS: Searching and Browsing with Personalization
- FRS: Update folder profiles
- SBS: Request new records for folder
- SBS: Searching and Browsing without Personalization

2.1.1.4 Content Management

Record management includes moving, copying and deleting records. Records are created while gathering records, they are not edited. In a similar manner queries can be saved during the search and browse phase, and later manipulated. Additional functionality in record handling consists in the possibility to rate records with regard to their relevance to the topic concerned, and to annotate records in the form of threaded discussions by the members of the community.

- CWS: Add record
- CWS: Annotate record

- CWS: Delete/cut record / query
- CWS: Move record / query
- CWS: Rate record
- SBS: Save query
- SBS: Save results

2.1.1.5 Working with Communities

The existence of communities in the Cyclades system is visible to all registered users of the system. While members of a community have access to the community's folders and their contents, nonmembers may only view some information of this community. Given this information, users may choose to subscribe a community, thus becoming a community member. It is also possible to create new communities that can then be populated by inviting users to join.

- CWS: Create a new community
- CWS: Delete a community
- CWS: Join a community via invitation
- CWS: Leave a community
- CWS: Manage communities
- CWS: View list of communities / Subscribe to a community

2.1.1.6 Awareness

Awareness is an important feature of a collaborative work environment like the CYCLADES system. This is supported by the means of email notifications and event icons. Event icons will be seen by the user while navigating the workspace. A user can edit her preferences for receiving these notifications about events, and can also catch up on recent events.

- CWS: Catch up
- CWS: Edit event notification

2.1.1.7 Communicate with Other Community Members

Communication with other community members is supported by discussion forums, where a member can create notes, and add notes to other members' threads.

- CWS: Create and add note

2.1.1.8 Collection Management

Collections can be created, edited and removed by members that have the necessary rights. Collections are subsets of the global information space that are meaningful for a specific group of users. The creator will usually browse the list of existing collections in order to find out whether a collection meeting the required characteristic is found. Each time a new archive is registered, a new collection is also created that maintains all the documents of the new archive.

- CS: Add Search/Browse Format
- CS: Create collection
- CS: Delete collection
- CS: Edit collection description
- CS: Remove Search/Browse Format
- SBS: browse attribute values
- SBS: browse folder collections
- SBS: browse personal collections
- SBS: browse system collections

2.1.1.9 Archive Management

The Cyclades system has to know about the archives it should gather records from, and for each archive, which metadata formats it should use. Furthermore, in order to define collections later on, the user might need a description of the archive content.

- AS: Register archive
- AS: Edit archive information
- AS: Delete archive

2.2 Access Service

- **Tester:** Sascha Kriewel, University of Duisburg-Essen (sascha.kriewel@uni-duisburg.de)
- **Test date:** April 7th, 2003
- **Scope of test:**
  - Registering a new archive
  - Editing archive information
  - Deleting an archive
- **Test environment:** Mozilla 1.2b [Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.2b) Gecko/20021016] on Debian Linux (Kernel 2.4)

2.2.1 Test plan

For testing the functionality of the Access Service part of the system we will be going through the three use cases of Archive Management listed above. For each use case we will perform the steps from logging into the system until completion that an actual user would have to perform, and verify the results.

- **Use Case:** Registering a new archive
  - call the Archive Management GUI
  - register a new archive

  - enter the URL of an OAI compliant repository
  - edit the archive information
  - submit the registration request

- **Use Case:** Editing archive information
  - call the Archive Management GUI
  - select an archive
  - edit the archive information
  - submit the changed information

- **Use Case:** Delete an archive
  - call the Archive Management GUI
  - delete an archive

2.2.2 Test log

```
Test Type : Functionality Test
Service   : Archive Management (Access Service)
Tester    : Sascha Kriewel, UniDuE
Email     : sascha.kriewel@uni-duisburg.de
Date      : April 7th, 2003

Use Case: Register a New Archive

- logged into the system
- opened Archive Management GUI
- choose "register new archive"
- entered a URL of an OAI compliant repository

  http://techreports.larc.nasa.gov/ltrs/oai2.0/

- added some information about the repository
- received a note acknowledging the registration
- new archive shows up in personal list of archives
- received an email that archive is indexed and searchable
- can now search in archive

[x] successfully completed

Use Case: Edit Archive Information

- logged into the system
- opened Archive Management GUI
- selected an archive from the list of owned archives
- edited the information that was presented and saved it
- selected the archive again and checked the changed information

[x] successfully completed

Use Case: Delete Archive
```

```
- logged into the system
- opened Archive Management GUI
- marked an archive from the list of owned archives
- deleted the archive

[x] successfully completed
```

2.2.3 Test results

All use cases previously defined could be successfully completed and had the expected results.

Two use cases that were tested as part of the use case tests for the Collection Service, *Add* and *remove a Search&Browse Format*, build on functionality of the Access Service that is not provided by the current prototype.

2.2.4 Test summary

All functionalities required from the Archive Management are provided and all associated use cases could be completed.

2.3 Collection Service

- **Tester:** M. Elena Renda {renda@iei.pi.cnr.it}
  Istituto di Scienza e Tecnologie della Informazione – CNR, Pisa
- **Test date:** April 9th, 2003
- **Scope of test:**
  - Selecting personal collection set
  - Becoming a collection administrator
  - Creating a new collection
  - Editing a collection description
  - Deleting a collection
  - Adding a Search&Browse Format (not supported)
  - Removing a Search&Browse Format (not supported)
- **Test environments:** Mozilla 1.3 [Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.3) Gecko/20030312] on Windows 2000

2.3.1 Test plan

For testing the functionality of the Collection Management part of the system we will be going through the use cases listed above.

For each use case we will perform the steps – from logging into the system until completion – that an actual user would have to perform.

- **Use Case:** Select personal collection set
  - call Collection Management GUI

  - choose the "Personal Collection set" option from the main menu of the Collection Management window
  - browse through and select collections from "All collections"
  - "Add" the selected collection(s)
  - "Submit" the choice

- **Use Case:** Become a collection administrator
  To become a collection administrator just create a new one; see the "Create a new collection" use case

- **Use Case:** Create a new collection
  - call Collection Management GUI
  - choose the "New" option from the "Collection" menu of the Collection Management window
  - fill in the form
  - confirm the new collection creation

- **Use Case:** Edit a collection description
  - call Collection Management GUI
  - select a collection from those created (owned)
  - choose the "Edit" option from the action menu of the Collection Management window
  - modify the description
  - confirm/discard the modification

- **Use Case:** Delete a collection
  - call Collection Management GUI
  - select a collection from those created (owned)
  - choose the "Delete" option from the action menu of the Collection Management window
  - confirm/discard the deletion

2.3.2 Test log

```
Test Type   : Functionality Test
Service     : Collection Service
Tester      : M.Elena Renda, I.S.T.I. - CNR
Email       : renda@iei.pi.cnr.it
Date        : April 9th, 2003

Use Case: Select personal collection set
- logged into the system
- opened Collection Management GUI
- selected the "Personal Collection set" option from the
  main menu of the Collection Management window
- browsed through and select a collection from "All collections"
- added the selected collection to the "Personal Collection set"
- submitted the choice

[X] successfully completed
```

```
Use Case: Create a new collection
- logged into the system
- opened Collection Management GUI
- selected the "New" option from the "Collection" menu of the Collection
  Management window
- filled in the form
- confirmed the new collection creation

[X] successfully completed


Use Case: Edit a collection description
- logged into the system
- opened Collection Management GUI
- selected a collection from those created (owned)
- chose "Edit" from the action menu of the Collection Management window
- modified the description form of the collection
- confirmed/discarded the modification

[X] successfully completed


Use Case: Delete a collection
- logged into the system
- opened Collection Management GUI
- selected a collection from those created (owned)
- chose "Delete" from the action menu of the Collection Management window
- confirmed/discarded the deletion

[X] successfully completed


Use Case: Add a Search&Browse Format
*** functionality not implemented, could not be tested ***

[E] not completed


Use Case: Remove a Search&Browse Format
*** functionality not implemented, could not be tested ***

[E] not completed
```

2.3.3 Test results

2.3.3.1 Use Cases that could be completed

All the use cases worked as specified, except for *Add/Remove Search&Browse Formats* ones.

2.3.3.2 Use Cases that didn't work

It was not possible to test the *Add/Remove Search&Browse Format* use cases. This functionality was not provided, and thus couldn't be tested.

2.3.4 Test summary

The main functionalities of the Collection Management part are provided and the associated use cases could be completed.

2.4 Collaborative Work Service

- **Tester:** Thomas Kreifelts, Fraunhofer-FIT (kreifelts@fit.fraunhofer.de)
- **Test date:** April 22-23, 2003
- **Scope of test:**
  - View and edit user information
  - Create folders and subfolders
  - Copy, move, delete, destroy folders
  - Associate or disassociate folder collections
  - Edit folder recommendation preferences
  - Move or copy records or queries
  - Delete records or queries
  - View communities and subscribe to a community
  - Join a community via invitation
  - Create a new community
  - Manage community membership and access rights
  - Leave a community
  - Rate records or queries
  - Annotate records or queries
  - Edit event notification preferences
  - Catch up
  - Create and add note to a discussion forum
- **Test environment:** Internet Explorer 5, Windows 2000

2.4.1 Test plan

For testing the functionality of the Collaborative Work Service we have gone through the use cases listed above.

- **View and edit user information:**

  View and edit personal preference settings
  - choose menu Options/Preferences
  - view personal preference settings including

    - permission to be recommended to other users
    - usage of Javascript, ActiveX
    - user interface language
    - user profile (expert, advanced, beginner)
  - edit personal preference settings (set your user profile to expert, so that all actions are shown in the menus)
  - hit OK

  View and edit personal information

  - choose menu Options/Details
  - view personal information including
    - name
    - organization
    - phone and fax numbers
    - image URL and home page
    - postal address
  - edit personal information
  - hit OK

- **Create folders and subfolders:**

  - choose menu File/New/{Community, Project, Private} Folder
  - fill in name, description of folder
  - specify collections to be associated to folder (only for root folders, subfolders inherit associated collections from their parent folders as default)
  - select recommendation preferences
  - hit OK

- **Copy, delete, destroy folders:**

  Copy folders

  - select folder(s) in the folder listing
  - hit copy in the multi-action bar
  - navigate to destination folder
  - choose menu Edit/Paste

  Delete folders

  - select folder(s) in the folder listing
  - hit delete in the multi-action bar

  Destroy folders

  - go to waste basket
  - select folder(s) in the folder listing
  - hit destroy in the multi-action bar

- **Associate or disassociate folder collections:**

  - choose {Add Collections, Remove Collections} in folder action menu
  - select collections to be associated/disassociated to or from folder

  - hit OK

- **Edit folder recommendation preferences:**
  - choose Edit Reco Prefs in folder action menu
  - set or reset recommendation preferences
  - hit OK

- **Move or copy records or queries:**
  - select records or queries in the folder listing
  - hit {cut/copy} in the multi-action bar
  - navigate to destination folder
  - choose menu Edit/Paste

- **Delete records or queries:**
  - select records or queries in the folder listing
  - hit delete in the multi-action bar

- **View communities and subscribe to a community:**
  - choose menu Goto/Communities (or hit Communities instant access navigation button)
  - view list of communities
  - choose Join in the action menu of a community that is open to subscription
  - view joined community's root folder

- **Join a community via invitation:**
  - choose menu Goto/Communities (or hit Communities instant access navigation button)
  - view list of communities
  - choose Mail Managers from the action menu of a community that is not open to subscription
  - request invitation by a community manager via e-mail
  - wait to be added as member to the community

- **Create a new community:**
  - choose menu File/New/Community in your home folder
  - fill in name and description of the new community, select the collections to be associated to the new community, set the recommendation preferences and choose whether the new community is open to subscription or not
  - hit OK

- **Manage community membership and access rights:**

  Invite new members (community managers only)
  - go to community root folder
  - hit the Invite Member button in the short-cut button list (if no short-cut buttons are shown, switch them on in the View menu)
  - select members to be added to the community, select role in which you want to add them (member or manager), and enter optional invitation text

  - hit OK

  Remove members (community managers only)

  - go to the community's member listing by hitting the Members button right next to the community root folder name in the location line
  - select members to remove in the member listing
  - hit remove in the multi-action bar

  Assign/revoke manager role (community managers only)

  - choose Assign Role in a community folder action menu
  - set or reset the role of the community members to or from manager
  - hit OK

- **Leave a community:**

  - go to your home folder
  - choose Delete in the community's action menu (you are still a member of the community)
  - go to the waste basket
  - choose Destroy in the community's action menu (the community is still intact for the remaining members)

- **Rate records or queries:**

  - select records or queries in the folder listing
  - hit rate in the multi-action bar
  - choose your ratings for the records and queries selected
  - hit OK

- **Annotate records or queries:**

  - choose Attach Note from the record or query action menu
  - select type of annotation, and subject and body of annotation
  - hit OK

- **Edit event notification preferences:**

  Edit default event notification preferences

  - choose menu Options/Default Events
  - set/reset your notification preferences per event type
  - hit OK

  Edit artifact specific event notification preferences

  - choose Events in the action menu of an artifact (folder, record, query)
  - set/reset your notification preferences per event type for this artifact (will override the default setting)
  - hit OK

- **Catch up:**

  - select folders, records, queries in a folder listing

  - hit catch up in the multi-action bar

- **Create and add note to a discussion forum:**
  - choose menu File/New/Discussion
  - enter name of discussion and first note by selecting type of note and entering its subject and body
  - hit OK

2.4.2 Test log

```
Test Type : Functionality Test
Service   : Collaborative Work Service
Tester    : Thomas Kreifelts
Email     : thomas.kreifelts@fit.fraunhofer.de
Date      : April 22-23, 2003

Use case: View and edit personal preference settings

- chose menu Options/Preferences
- was presented with personal preference settings including:
     permission to be recommended to other users,
     usage of Javascript, ActiveX,
     user interface language,
     user profile (expert, advanced, beginner)
- set personal preferences:
     allowed my recommendation to others
     user profile to expert

[x] successfully completed

Use case: View and edit personal information

- chose menu Options/Details
- was presented with current personal information
- edited personal information by completing full name and
  entering organization and image URL

[x] successfully completed

Use case: Create private root folder

- chose menu File/New/Private Folder in home folder
- was presented with folder creation form
- filled in name, description of folder
- specified collections to be associated to folder
- selected recommendation preferences for folder
- was presented with home folder listing including
  the new private root folder

[x] successfully completed

Use case: Create community subfolder
```

```
- chose menu File/New/Community Folder in a community folder
- was presented with folder creation form
- filled in name, description of folder
- selected recommendation preferences for folder
- was presented with community folder listing including
  the new subfolder

[x] successfully completed

Use case: Copy folders

- selected folders in the folder listing
- hit copy in the multi-action bar
- navigated to destination folder
- chose menu Edit/Paste
- was presented with destination folder listing including
  the copied folders

[x] successfully completed

Use case: Delete folders

- selected folder in a folder listing
- hit delete in the multi-action bar
- was presented with a folder listing where the deleted
  folders were missing
- gone to waste basket where the deleted folders were present

[x] successfully completed

Use case: Destroy folders

- gone to waste basket
- selected folders in the waste listing
- hit destroy in the multi-action bar
- was presented with a waste listing where the destroyed
  folders were missing

[x] successfully completed

Use case: Associate folder collections

- chose Add Collections in a folder action menu
- was presented with a list of collections from personal
  favourite collections not yet associated to folder
- selected two additional collections to associate
- gone to folder info page where these collections were shown
  as associated to the folder

[x] successfully completed

Use case: Disassociate folder collections

- chose Remove Collections in a folder action menu
```

```
- was presented with the list of collections currently
  associated to the folder
- selected one collection to be disassociated from folder
- gone to the folder info page where this collection was no
  longer shown as associated to the folder

[x] successfully completed

Use case: Edit folder recommendation preferences

- chose Edit Reco Prefs in folder action menu
- set recommendation preferences to Users, Communities
- gone to the folder info page where these preferences were shown

[x] successfully completed

Use case: Move records

- selected two records in a community folder listing
- hit cut in the multi-action bar
- was presented with the folder listing where the two records
  were missing
- navigated to the parent folder
- chose menu Edit/Paste
- was presented with the parent folder listing including the
  two records

[x] successfully completed

Use case: Copy queries

- selected a query in a private folder listing
- hit copy in the multi-action bar
- was presented with an unchanged folder listing
- navigated to a subfolder
- chose menu Edit/Paste
- was presented with the subfolder listing including the query

[x] successfully completed

Use case: Delete records

- selected two records in a private folder listing
- hit delete in the multi-action bar
- was presented with a folder listing were both records were
  missing
- went to the waste basket where both records were shown

[x] successfully completed

Use case: View communities and subscribe to a community

- chose menu Goto/Communities
- was presented with the list of current communities
```

```
- chose Join from the action menu of community 'Test Community'
  which is open to subscription
- was presented with 'Test Community' root folder

[x] successfully completed

Use case: Join a community via invitation

- chose menu Goto/Communities
- was presented with the list of current communities
- chose Mail Managers from the action menu of community
  'Closed Shop' which is not open to subscription
- was presented with a message template of the local mail agent
  where the community managers of 'Closed Shop' were preset as
  receivers
- sent a request to the managers and eventually got invited

Remark: The body of the invitation message was garbled (wrong
        encoding). This is a bug of the underlying BSCW version
        4.1.1. With transition to version 4.1.4 this bug will go
        away.

[x] successfully completed with minor bug

Use case: Create a new community

- chose menu File/New/Community in home folder
- was presented with a community creation form
- filled in name and description of the new community
- selected the collections to be associated to the new community
- set the recommendation preferences
- set the new community to be open to subscription
- was presented with a home folder listing also showing the
  new community

[x] successfully completed

Use case: Invite new community members

- gone to community root folder
- hit the Invite Member button in the short-cut button list
- selected two members to be added to the community
- set the member role to Member
- gone to the community root folder info page where the new
  members were shown

Remark: The body of the invitation message was garbled (wrong
        encoding). This is a bug of the underlying BSCW version
        4.1.1. With transition to version 4.1.4 this bug will go
        away.

[x] successfully completed with minor bug

Use case: Remove community members
```

```
- hit the Members button in the location line showing the
  community
- was presented with the community's member listing also
  indicating current managers
- selected two members to be removed from the community
- hit remove in the multi-action bar
- gone to the folder info page where these two members were no
  longer shown

[x] successfully completed

Use case: Assign/revoke manager role

- chose Assign Role in a community folder action menu
- was presented with a form where for all current members of the
  community the present role was indicated
- set the manager role for one member and switched back a current
  manager to (ordinary) member
- gone to the folder info page where these new role assignments
  were shown

[x] successfully completed

Use case: Leave a community

- gone to home folder
- chose Delete in the community's action menu
- gone to the waste basket containing the deleted community
- chose Destroy in the community's action menu

[x] successfully completed

Use case: Rate records

- selected three records in a folder listing
- hit rate in the multi-action bar
- was presented with a form indicating my current ratings if any
  for the records selected
- changed the rating for one record
- gone to the record info page where the recent rating was shown

[x] successfully completed

Use case: Annotate queries

- chose Attach Note from the query action menu
- selected Pro as type of annotation, and entered subject and
  body of annotation
- was presented with a folder listing where the annotation was
  indicated by an icon in the Note column

[x] successfully completed
```

```
Use case: Edit default event notification preferences

- chose menu Options/Default Events
- set Daily report notification preference for create events
- received an activity report on creation of new artifacts the
  day after

[x] successfully completed

Use case: Edit artifact specific event notification preferences

- chose Events in the action menu of a record
- reset the Event Icon notification preferenc for Change Events
- subsequent rating of record produced no more change event icons

[x] successfully completed

Use case: Catch up


- selected a subfolder and two records in a folder listing
- hit catch up in the multi-action bar
- event icons for subfolder and both records disappeared,
  which was also true for the contents of the subfolder

[x] successfully completed

Use case: Create and add note to a discussion forum

- chose menu File/New/Discussion
- entered name of discussion and first note by selecting type
  Note and entering its subject and body
- was presented with a folder listing that included the newly
  created discussion forum

[x] successfully completed
```

2.4.3 Test results

All use cases worked as previously defined with the exception of a minor bug (wrong encoding used for invitation messages) which occured in the use cases *Join a community via invitation* and *Invite new community members*. This bug will be removed by upgrading to BSCW 4.1.4 as the underlying CWS platform.

2.4.4 Test summary

The functionality of the Collaborative Work Service is provided and the list of use cases could essentially be completed as specified in D3.0.1.

2.5 Filtering and Recommendation Service

- **Tester:** M. Elena Renda {renda@iei.pi.cnr.it}

Istituto di Scienza e Tecnologie della Informazione – CNR, Pisa

- **Test date:** April 9th, 2003
- **Scope of test:**
  - Searching and Browsing with Personalization
  - Updating folder profiles
  - Recommending collections for folder
  - Recommending communities for folder
  - Recommending records for folder
  - Recommending users for folder
- **Test environments:** Mozilla 1.3 [Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.3) Gecko/20030312] on Windows 2000

2.5.1 Test plan

For testing the functionalities of the Filtering and Recommendation part of the system we will be going through the use cases listed above.

For each use case we will perform the steps – from logging into the system until completion – that an actual user would have to perform.

- **Use Case:** Search with Personalization

  Search on-demand
  - enter a folder
  - call Search and Browse GUI
  - choose the "Action − > get new records for folder" option from the main menu of the Search and Browse window

  Search ad-hoc
  - enter a folder
  - call Search and Browse GUI
  - select a query from history or folder if any
  - choose the "Current query − > Submit personalized" option from the main menu of the Search and Browse window
- **Use Case:** Update folder profile
  - enter a personal folder
  - choose the "Update Folder Profile" option from the folder menu
- **Use Case:** Obtain recommendations for folder (Records, Users, Collections, Communities)
  - choose the "Edit Reco Prefs" option from the folder menu
  - choose the option to obtain the recommendations desired

2.5.2 Test log

```
Test Type   : Functionality Test
Service     : Filtering and Recommendation Service
Tester      : M.Elena Renda, I.S.T.I. - CNR
Email       : renda@iei.pi.cnr.it
Date        : April 9th, 2003


Use Case: Search with Personalization

(a)  Search on-demand
- logged into the system
- entered the folder "DL" - opened Search and Browse GUI
- selected the "get new records for folder" option from "Action" in the Search
  and Browse window
- obtained 21 records, 15 of which pertinent to the folder
- 1 minute later, selected again the "get new records for folder" option from
  "Action" in the Search&Browse window
- as expected, the system returned "no records found"

[X] successfully completed


(b) Search ad-hoc - logged into the system
- entered the folder "DL"
- opened Search&Browse GUI
- selected a query from the history of the folder
- chosen the "Submit personalized" option from "Current query"  in the
  main menu of the Search&Browse window
- obtained 6 over 9 fitting records

[X] successfully completed


Use Case: Update folder profile
- logged into the system
- entered a folder
- opened Search&Browse GUI
- chosen the "Update Folder Profile" option from the folder menu
- confirmed request for updating the folder profile

[X] successfully completed


Use Case: Obtain recommendations for folder
- logged into the system
- entered the folder "Digital Library"
- chosen the "Edit Reco Prefs" option from the folder menu
- chosen the option to obtain all recommendations (Records, Users, Collections,
Communities)


(a) Users
```

```
- obtained 4 recommendations of users

[X] successfully completed


(b) Collections
- obtained 13 recommendations of collections

[X] successfully completed


(c) Communities
- obtained 3 recommendations of communities

[X] successfully completed


(d) Records
- obtained 13 recommendations of records

[X] successfully completed
```

2.5.3 Test results

2.5.3.1 Use Cases that could be completed

All the use cases worked as specified.

2.5.3.2 Use Cases that didn't work

There are no use cases that could not be completed

2.5.4 Test summary

The main functionalities of the Filtering and Recommendation part are provided and all the associated use cases could be completed.

2.6 Mediator Service

- **Tester:** Thomas Kreifelts, Fraunhofer-FIT (thomas.kreifelts@fit.fraunhofer.de)
- **Test date:** April 16, 2003
- **Scope of test:**
  - Register as a new user
  - Login
  - Change password
  - Un-register from system
  - Manage registration rights
- **Test environment:** Internet Explorer 5, Windows 2000

2.6.1 Test plan

For testing the functionality of the Mediator Service we have gone through the use cases listed above.

- **Register as a new user:**
  - go to registration page
  - enter name, password, e-mail address
- **Login:**
  - go to login page
  - enter user name and password
- **Change password:**
  - login to the system
  - select 'change password' menu option
  - enter new password
- **Un-register from system:**
  - login to the system
  - select 'un-register' menu option
- **Manage registration rights:**
  - login to the system as administrator
  - select 'registration management' menu option
  - set or reset a user's right to register archives and/or create collections

2.6.2 Test log

```
Test Type : Functionality Test
Service   : Mediator Service
Tester    : Thomas Kreifelts
Email     : thomas.kreifelts@fit.fraunhofer.de
Date      : April 16, 2003

Use case: Register as a new user

- gone to registration page
- entered user name, password and e-mail address
- welcomed as new user
- continued to be presented with home folder

[x] successfully completed

Use case: Login

- gone to login page
- entered name and password
- was presented with home folder
```

```
Remark: Access to the folder system required
second authentication.

[x] successfully completed


Use case: Change password

- logged into the system
- selected ‘Change password’ option from the ‘Options’ menu
- entered old and new passwords

Remark: Access to the folder system required renewed
        authentication.

[x] successfully completed


Use case: Un-register from system

- logged into the system
- selected ‘Unregister’ option from the ‘Options’ menu
- confirmed unregistration with password
- couldn’t access folders anymore (as expected)

Remarks:
1) This is new functionality not described in D3.0.1.
2) Took about 10 seconds.
3) After un-registration, Collection and Archive Services were
   still shown in the main menu; hitting this options, however,
   produced a NullPointerException.

[x] successfully completed

Use case: Manage registration rights

*** functionality missing, could not be tested ***

This use case is about managing the rights to register new
archives and create new collections. With the present system,
any user can register new archives and create new collections.

[E] not completed
```

2.6.3 Test results

2.6.3.1 Use cases that could be completed

The use cases for *Registration*, for *Login*, and for *Change password* worked as previously defined. The use case for *Un-registration* worked as might be expected.

2.6.3.2 Use cases that didn't work

It was not possible to *manage registration rights*. This functionality was not provided, and thus couldn't be tested. The original use case covered the management of the right to register new archives and the right to create new collections. In the present system, it is not necessary to manage these rights, since any user may register new archives and may create new collections.

2.6.4 Test summary

The main functionality of the Mediator Service is provided and the associated use cases could be completed. The use case that could not be successfully tested is obsolete with the present version of the system.

The Mediator service provides an additional set of functions for a system administrator. These functions have not been covered as use cases in D3.0.1, and hence not been included in the present functionality test. They were, however, successfully tested as reported in D4.2.1.

2.7 Search and Browse Service

- **Tester:** Sascha Kriewel, University of Duisburg-Essen (sascha.kriewel@uni-duisburg.de)
- **Test date:** April 2nd, 2003
- **Scope of test:**
  - Browsing system collections
  - Browsing folder collections
  - Browsing personal collections
  - Browsing attribute values
  - Requesting new records for folder
  - Saving results
  - Saving queries
  - Submitting query without personalization
- **Test environment:** Mozilla 1.2b [Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.2b) Gecko/20021016] on Debian Linux (Kernel 2.4)

2.7.1 Test plan

For testing the functionality of the Search and Browse part of the system we will be going through the use cases listed above. For each use case we will perform the steps from logging into the system until completion that an actual user would have to perform.

The Use Case "Saving results" will be tested as part of "Get New Records For Folder", as it only makes sense in the context of a result list of records.

- **Use Case:** Search without Personalization
  - enter a personal folder
  - call Search and Browse GUI
  - select a query from history or folder if any

  - (browse through and select collections)[1]
  - edit the query
  - submit the query
  - view a record

- **Use Case:** Browse Attribute Values
  - enter a personal folder
  - call Search and Browse GUI
  - select a metadata schema
  - browse attribute values
  - select a value for current query

- **Use Case:** Request New Records for Folder
  - enter a personal folder with records and associated collections
  - call Search and Browse GUI
  - fetch new records for folder
  - view the records
  - save the records to folder if so inclined

- **Use Case:** Browse Folder Collections
  - enter a personal folder with associated collections
  - open Search and Browse GUI
  - browse folder collections
  - view collection information
  - maybe select collection(s) for current query
  - quit browsing

- **Use Case:** Browse Personal Collections
  - enter a personal folder
  - open Search and Browse GUI
  - browse personal collections
  - view collection information
  - maybe select collection(s) for current query
  - quit browsing

- **Use Case:** Browse System Collections
  - enter a personal folder
  - open Search and Browse GUI
  - browse system collections
  - view collection information
  - maybe select collection(s) for current query
  - quit browsing

[1] this will be tested seperately

- **Use Case:** Save query
  - enter a folder
  - open Search and Browse GUI
  - add conditions to query
  - add collections to query
  - save query to folder
  - clear selected conditions and collections
  - select query from folder

2.7.2 Test log

```
Test Type : Functionality Test
Service   : Search and Browse Service
Tester    : Sascha Kriewel, UniDuE
Email     : sascha.kriewel@uni-duisburg.de
Date      : April 2nd, 2003

Use Case: Search without Personalization

- logged into the system
- entered a folder
- opened Search and Browse GUI
- selected a previously sent query
- used the interface to edit the query and construct a
  query of several conditions
- send the query and got a list of appropriate records
- selected a result and received the actual record

[x] successfully completed

Use Case: Browse Attribute Values

- logged into the system
- entered a folder
- opened Search and Browse GUI
- started a new query
- added an empty condition
- selected field 'subject' for the new condition
- called 'Show Values' from the condition menu
- browsed within values
- closed attribute value window

[x] successfully completed

Use Case: Request New Records for Folder / Save Records

- logged into the system
- created a new folder "Women Literature", associated with
  one collection: celebration
- entered folder
- opened Search and Browse GUI
```

```
- searched and added appropriate records to the folder
- called "Get New Records For Folder" from menu
- received four fitting records:

o Juliana Horatia Ewing and Her Books
o Leaves from Juliana Horatia Ewing's "Canada Home"
o Letters and Memorials of Jane Welsh Carlyle
o New Letters and Memorials of Jane Welsh Carlyle

- saved records to folder
- called "Get New Records For Folder" from menu
- received no more records as expected
- refreshed folder and got the saved folder

[x] successfully completed

Use Case: Browse Folder Collections

- logged into the system
- entered previously created folder: "Women Literature"
- opened Search and Browse GUI
- called "Browse folder collections" from menu
- received a list consisting of the collection previously
  associated with this folder
- chose a collection to display details
- closed the information window
- selected the collection and added it to the
  current query

[x] successfully completed

Use Case: Browse Personal Collections

- logged into the system
- entered a folder
- opened Collection Management GUI
- defined a set of personal collections
- submitted the set
- opened Search and Browse GUI
- later called "Browse personal collections" from menu
- received a list consisting of all the collections in my
  personal set
- chose a collection to display details
- closed the information window
- selected all collections and added them to the current
  query

[x] successfully completed

Use Case: Browse System Collections

- logged into the system
- entered previously created folder: "Women Literature"
- opened Search and Browse GUI
```

```
- called "Browse system collections" from menu
- received a complete list consisting of all collections
  known to the system
- chose a collection to display details
- closed the information window
- selected some collections and added them to the
  current query

[x] successfully completed

Use Case: Save query

- logged into the system
- entered a folder
- opened Search and Browse GUI
- added three conditions to empty query:

 +, subject, contains, information
 +, creator, !=, Miller
 +, language, =, eng

- added collections from system to query
- saved query to folder as ‘‘Test Query’’
- refreshed folder and selected ‘‘Test Query’’
- verified query

[x] successfully completed
```

2.7.3 Test results

2.7.3.1 Use Cases that could be completed

All use cases worked as previously defined.

2.7.3.2 Use Cases that didn't work

None.

2.7.4 Test summary

All functionalities of the Search and Browse service are provided and the associated use cases could be completed.

2.8 Summary

During the specification phase of WP2 a number of use cases were identified for the CYCLADES system. These were use cases for single users, communities, projects, collection and archive management, and for manipulating archive/collection registration rights. In a series of functionality tests it was verified that all these use cases could be completed with the current prototype implementation of the CYCLADES system.

In summary it can be said that nearly all requirements of the specification phase have been met by the CYCLADES prototype, with two notable exceptions. The given system does not provide facilities for managing registration rights of users, so that every user can register new archives and create collections. This was a conscious design decision.

There is also missing the functionality to choose search and browse formats for a given collection of records. For each existing collection the collection owner was supposed to be able to define available metadata formats for searching and browsing. Uses cases for adding and removing search&browse formats were therefore specified, but are not available with the current system.

Aside from these, all functionality specified in the Global System Architecture Report (D2.2.1) and the System Specification Report (D3.0.1) is provided. All the use cases that were tested could be successfully completed by a user.

Chapter 3

Efficiency Evaluation

3.1 Introduction

The aim of efficiency testing of the CYCLADES prototype was to quantify the performance of the various components as well as the overall system. Towards this end we determined

- average response time to user requests,
- system scalability, and
- system stability.

The efficiency testing was split into two phases. During the first phase the performance of the implemented methods was tested, using an XML-RPC test client. The methods were subjected to stress tests by forcing them to handle an increasing number of concurrent requests at once, and also by testing them with large amount of data. By this a better view of the scalability of implemented prototype was to be gained.

In a similar manner, the system as a whole was tested for its performance, by going through all defined use cases, and noting the efficiency with which the system carried them out, but also the scalabilty of the system regarding large numbers of user requests and large amounts of data. Lastly, it was also determined how the CYCLADES prototype fares with regards to its stability when one or several of its service components are unavailable.

3.2 API

3.2.1 Access Service

- **Tester:** Sascha Kriewel, University of Duisburg-Essen (sascha.kriewel@uni-duisburg.de)
  Saadia Malik, University of Duisburg-Essen (malik@is.informatik.uni-duisburg.de)
- **Test date:** April 8th, 2003
- **Scope of test:**
  - *deleteArchive*
  - *deleteUser*
  - *forgetTempForUser*

- *getArchiveDescription*
- *getArchiveForUser*
- *getArchives*
- *getAttributeValues*
- *getAttributes*
- *getId*
- *getIndexedTermsAndWeight*
- *getMetadataAttributeTerms*
- *getRecords*
- *getSchemaForArchives*
- *registerArchive*
- *saveArchive*
- *search*

- **Test environment:** JMeter 1.8.1, Mozilla 1.2b [Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.2b) Gecko/20021016] on Debian Linux (Kernel 2.4)

3.2.1.1 Test plan

To measure the API efficiency, each method call was simulated using the XML-RPC with the help of the JMeter application. For each method, a test plan was made that is composed of an XML document containing the method name and the parameters passed, and a thread group containing the information of concurrent calls sent and number of times these cocurrent calls are sent. The following numbers of concurrent calls are sent to each method: 1, 5, 10, 15, 20. Each of these concurrency runs is sent several times to determine an average result. During this, method response time and errors are observerved.

3.2.1.2 Test log

```
Test 1:

Method:
  getId()

Parameters:
  none

Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------
    1      |  200   |  200   |    476ms        |    x
    5      |   40   |  200   |   1667ms ~ 2s   |    x
   10      |   20   |  200   |   2081ms ~ 2s   |    x
   15      |   15   |  225   |   2449ms ~ 2s   |    x
   20      |   10   |  200   |   3608ms ~ 4s   |    x
```

```
Test 2:

Method:
  getIndexedTermsAndWeights()

Parameters:
  list of recordId
  maxTermNo

Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |  200   |  200   |   1041ms ~ 1s   |    x
    5      |   40   |  200   |   3191ms ~ 3s   |    x
   10      |   20   |  200   |   5268ms ~ 5s   |    x
   15      |   15   |  225   |   7033ms ~ 7s   |    x
   20      |   10   |  200   |   8810ms ~ 9s   |    x



Test 3:

Method:
  search()

Parameters:
  query
  maxRecordNo
  maxTermNo
  timeStamp

Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |  200   |  200   |   4201ms ~ 4s   |    x
    5      |   40   |  200   |  13124ms ~ 13s  |  98%
   10      |   20   |  200   |  24313ms ~ 24s  |  97%
   15      |   15   |  225   |  34276ms ~ 34s  |  90%
   20      |   10   |  200   |  45379ms ~ 45s  |  93%



Test 4:

Method:
  getArchiveDescription()

Parameters:
  oaiId
```

Average Response Time:

concurrent requests	loops	total runs	average response-time	results o.k.?
1	200	200	445ms	x
5	40	200	1816ms ~ 2s	x
10	20	200	2245ms ~ 2s	x
15	15	225	2482ms ~ 2s	x
20	10	200	2813ms ~ 3s	x

Test 5:

Method:
deleteArchive()

Parameters:
archiveId
owner

Description of Test:
see Testplan

Average Response Time:

concurrent requests	loops	total runs	average response-time	results o.k.?
1	200	200	945ms ~ 1s	x
5	40	200	1873ms ~ 2s	x
10	20	200	3615ms ~ 4s	x
15	15	225	4307ms ~ 4s	x
20	10	200	4739ms ~ 5s	x

Test 6:

Method:
forgetTempForUser()

Parameters:
owner

Average Response Time:

concurrent requests	loops	total runs	average response-time	results o.k.?
1	200	200	452ms	x
5	40	200	1205ms ~ 1s	x
10	20	200	1848ms ~ 2s	x

```
   15      |   15   |  225   |   2154ms ~ 2s   |    x
   20      |   10   |  200   |   2307ms ~ 2s   |    x
```

Test 7:

Method:
  getRecords()

Parameters:
  list of recordIds

Average Response Time:

```
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
 ------------------------------------------------------------
    1      |  200   |  200   |     417ms       |    x
    5      |   40   |  200   |    1578ms ~ 2s  |    x
   10      |   20   |  200   |    2024ms ~ 2s  |    x
   15      |   15   |  225   |    2468ms ~ 2s  |    x
   20      |   10   |  200   |    2975ms ~ 3s  |    x
```

Test 8:

Method:
  deleteUser()

Parameters:
  userId

Average Response Time:

```
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
------------------------------------------------------------
    1      |  200   |  200   |    461ms        |    x
    5      |   40   |  200   |   1203ms ~ 1s   |    x
   10      |   20   |  200   |   2019ms ~ 2s   |    x
   15      |   15   |  225   |   2241ms ~ 2s   |    x
   20      |   10   |  200   |   2838ms ~ 3s   |    x
```

Test 9:

Method: getArchives()

Description: This method exports the registered archives

Average Response Time:

concurrent requests	loops	total runs	average response-time	results o.k.?
1	20	20	142 ms < 1 s	x
5	20	100	453 ms < 1 s	x
10	20	200	752 ms ~ 1 s	x
15	20	300	1166 ms ~ 1 s	x
20	20	400	1509 ms ~ 1 s	x

concurrent requests	loops	total runs	average response-time	results o.k.?
1	100	100	156 ms < 1 s	x
5	40	200	485 ms < 1 s	x
10	20	200	767 ms ~ 1 s	x
15	15	225	1086 ms ~ 1 s	x
20	19	200	1442 ms ~ 1 s	x

Test 10:

Method: getArchiveForUser

Description: gets the archive that are owned by user

Parameter:
Input: owner
Output: List of Archive Objects

Average Response Time:

concurrent requests	loops	total runs	average response-time	results o.k.?
1	20	20	51 ms < 1 s	x
5	20	100	248 ms < 1 s	x
10	20	200	440 ms < 1 s	x
15	20	300	694 ms ~ 1 s	x
20	20	400	742 ms ~ 1 s	x

Test 11:

Method: registerArchive

Description: creates a new archive object and collects the initial
data from the data provider at url and remembers the user owner as the
owner of this archive

Parameter:
Input: url
owner
Output: A new archive object

Average Response Time:

concurrent requests	loops	total runs	average response-time	results o.k.?
1	200	200	665ms	x
5	40	200	734ms	98%
10	20	200	1045ms	90%

Test 12:

Method: saveArchive()

Description: saves the archive to persistent storage

Parameters:
Input: Archive

Description of Test:
see Testplan

Average Response Time:

concurrent requests	loops	total runs	average response-time	results o.k.?
1	20	20	66 ms < 1s	x
5	20	100	302 ms < 1s	x
10	20	200	603 ms ~ 1 s	x
15	20	300	970 ms ~ 1 s	x
20	20	400	1401 ms ~ 1 s	x

Test 13.1:

Method: getSchemaForArchives (parameterized)

Description: This method exports the list of schemas that the
specified archives supply (if no archive ids are specified, then all
the metadata schemas of all archives are listed)

Parameters:
Input: list of ArchiveId
Output: list of Schemas

Average Response Time:

concurrent requests	loops	total runs	average response-time	results o.k.?
1	20	20	241 ms ~ s	x
5	20	100	894 ms ~ 1 s	x

```
   10      |   20   |  200   |  1626 ms ~ 2 s  |  x
   15      |   20   |  300   |  2624 ms ~ 3 s  |  x
   20      |   20   |  400   |  3578 ms ~ 4 s  |  x


Test 13.2:

Method:  getSchemaForArchives (without parameters )

Description: This method exports the list of schemas that the
specified archives supply (if no archive ids are specified, then all
the metadata schemas of all archives are listed)

Parameters:
Input: list of ArchiveId
Output: list of Schemas

Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |   20   |   20   |   262 ms < 1 s  |  x
    5      |   20   |  100   |   749 ms ~ 1 s  |  x
   10      |   20   |  200   |  1551 ms ~ 2 s  |  x
   15      |   20   |  300   |  2621 ms ~ 3 s  |  x
   20      |   20   |  400   |  3990 ms ~ 4 s  |  x


Test 14:

Method:  getAttributes

Description: this method returns the list of attributes that the
Schema schemaName contains

Parameters:
Input: schemaName
Output: list of AttributeObjects

Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |   20   |   20   |   185 ms ~   s  |  x
    5      |   20   |  100   |   723 ms ~ 1 s  |  x
   10      |   20   |  200   |  1166 ms ~ 1 s  |  x
   15      |   20   |  300   |  1661 ms ~ 2 s  |  x
   20      |   20   |  400   |  2348 ms ~ 2 s  |  x

Test 15:

Method:  getAttributeValues
```

```
Description: this method returns the list of attributes that the
Schema schemaName contains

Parameters:
Input: archiveId
       schemaName
       attributeName
       maxNo
Output: a list of values, their type according to the type of attribute

Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |   20   |   20   |   371 ms        |  x
    5      |   20   |  100   |  1074 ms ~ 1s   |  x
   10      |   20   |  200   |  1892 ms ~ 2s   |  x
   15      |   20   |  300   |  2957 ms ~ 3s   |  x
   20      |   20   |  400   |  4057 ms ~ 4s   |  x


Test 16:

Method:  getMetadataAttributeTerms

Description:  this method returns indexed terms with their weights

Parameters:
Input: archiveId
       schemaName
       attributeName
       maxNo
Output: a list of pairs(term,weight)

Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |   20   |   20   |   317 ms        |  x
    5      |   20   |  100   |   621 ms ~ 1 s  |  x
   10      |   20   |  200   |  1131 ms ~ 1 s  |  x
   15      |   20   |  300   |  1677 ms ~ 2 s  |  x
   20      |   20   |  400   |  2227 ms ~ 2 s  |  x
```

3.2.1.3 Test results and summary

All the methods are working very efficiently with the exception of *search*. This method threw exceptions under heavy load. Otherwise overall average response time didn't exceed 4 seconds.

3.2.2 Collection Service

- **Testers:** Leonardo Candela, M.Elena Renda

  Istituto di Scienza e Tecnologie della Informazione – CNR, Pisa

  {candela, renda}@iei.pi.cnr.it

- **Test date: April 15th, 2003**

- **Scope of test:**

  - *addCollection*
  - *initializeCollection*
  - *listCollections*
  - *getPersonalCollections*
  - *editCollection*
  - *getCollectionMetadata*
  - *deleteCollection*
  - *deleteUser*
  - *deleteArchive*

- **Test environment:** Apache JMeter 1.8.1, Mozilla 1.3 [Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.3) Gecko/20030312] on Windows 2000

3.2.2.1 Test plan

For each method of the API of the Collection Service a test plan was created with the Apache JMeter test kit, simulating an XML-RPC request to the system. For these test suites it was determined how fast the system responded to each request. After that for each API call the system was subjected to a critical evaluation of its performance under high load, to determine its scalability. Therefore the test suits were started concurrently with an increasing number of simultaneous runs. Again the response times were determined. We report the min, the max and the average response time, and the error percentage of all tests. It was also checked how reliable the system was under the high load, by verifying the results and noting errors.

3.2.2.2 Test log

In the tables that follow are reported the results of all API tests, in terms of response time and error percentage.

Method: addCollection
Parameters: ()
Results:

concurrent calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	166	130	1091	0,00%	6,1/sec
5	40	200	685	160	1442	0,00%	6,6/sec
10	20	200	1166	140	2774	0,00%	6,1/sec
15	15	225	9797	541	38085	0,00%	1,4/sec
20	10	200	2571	150	5779	0,00%	5,5/sec

Method: initializeCollections
Parameters: collectionId
collectionName
collectionDescription
membershipCondition
userId
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	169	120	2444	0,00%	6,3/sec
5	40	200	727	130	5308	0,00%	6,2/sec
10	20	200	1700	140	12097	0,00%	4,9/sec
15	15	225	8884	30	24145	0,00%	1,6/sec
20	10	200	855	10	7841	0,00%	5,8/sec

Method: listCollections
Parameters: ()
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	111	80	441	0,00%	9,0/sec
5	40	200	395	80	1031	0,00%	10,3/sec
10	20	200	568	90	1822	0,00%	9,9/sec
15	15	225	4615	10	7651	0,00%	2,9/sec
20	10	200	230	90	1742	0,00%	7,5/sec

Method: listCollections
Parameters: userId
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	154	120	531	0,00%	6,5/sec
5	40	200	621	130	1722	0,00%	6,9/sec
10	20	200	1017	140	2363	0,00%	6,8/sec
15	15	225	5698	30	10766	0,00%	2,3/sec
20	10	200	879	140	2914	0,00%	6,1/sec

Method: getPersonalCollections
Parameters: userId
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	151	130	620	0,00%	6,6/sec
5	40	200	647	130	1262	0,00%	6,6/sec
10	20	200	1066	140	2664	0,00%	6,7/sec
15	15	225	1110	140	2772	0,00%	6,1/sec
20	10	200	1220	150	2824	0,00%	5,8/sec

Method: editCollection
Parameters: collectionMetadata
userId
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	346	290	1092	0,00%	2,9/sec
5	40	200	1652	171	3144	0,00%	2,8/sec
10	20	200	3149	350	5729	0,00%	2,8/sec
15	15	225	5107	521	8022	0,00%	2,6/sec
20	10	200	5920	601	12338	0,00%	2,4/sec

Method: getCollectionMetadata
Parameters: collectionIds
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	272	240	881	0,00%	3,7/sec
5	40	200	1353	260	2403	0,00%	3,5/sec
10	20	200	2653	260	4857	0,00%	3,2/sec
15	15	225	4010	391	8612	0,00%	3,3/sec
20	10	200	1659	270	5378	0,00%	4,5/sec

Method: deleteCollection
Parameters: collectionId
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	91	60	1022	0,00%	11,5/sec
5	40	200	726	110	5888	0,00%	5,6/sec
10	20	200	608	70	4396	0,00%	10,9/sec
15	15	225	1835	130	8111	0,00%	6,3/sec
20	10	200	895	80	8802	0,00%	6,5/sec

Method: deleteArchive
Parameters: archiveId
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	232	110	4647	0,00%	4,8/sec
5	40	200	1355	241	24756	0,00%	4,8/sec
10	20	200	3501	10	54057	0,00%	4,6/sec
15	15	225	6131	40	117780	0,00%	1,7/sec
20	10	200	851	120	3615	0,00%	6,5/sec

Method: deleteUser
Parameters: userId
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	85	70	410	0,00%	11,7/sec
5	40	200	314	70	962	0,00%	12,3/sec
10	20	200	703	90	1853	0,00%	9,1/sec
15	15	225	838	70	2634	0,00%	10,5/sec
20	10	200	91	70	180	0,00%	8,1/sec

3.2.2.3 Test results and summary

Generally, the system showed efficiency in handling all the tested APIs by responding to all the requests within short time. The system responded to all API calls without errors.

3.2.3 Collaborative Work Service

- **Tester:** Wido Wirsam, Fraunhofer FIT (wido.wirsam@fit.fraunhofer.de)
- **Test date:** April 30th, 2003
- **Scope of test:**
  - *updatePasswd*
  - *getFolders*
  - *getName*
  - *getDescription*
  - *getMembers*
  - *getQueries*
  - *getParents*
  - *getChildren*
  - *getCollections*
  - *getCommunity*
  - *saveQuery*
  - *saveResults*
  - *saveRecommendedRecords*
  - *saveRecommendedUsers*
  - *saveRecommendedCommunities*
  - *saveRecommendedCollections*
  - *addModifyCollection*
  - *updatePersonalCollections*
- **Test environment:** JMeter 1.8.1, on Windows XP

3.2.3.1 Test plan

To measure the API efficiency, each method call was simulated using the XML-RPC interface. The simultanous execution was performed using the JMeter application. For each method, a test plan has been created with the methods name and parameters encoded in XML. Each method was called with the following number of concurrent threads: 1, 5, 10, 20. Those calls have been repeated until the total number of 100 calls of every method was reached to determine an average result. During this, method response times and errors were observed.

3.2.3.2 Test log

```
Test 1:

Method:
  updatePasswd()

Parameters:
  name, password


Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |  100   |  100   |    401ms        |    o.k.
    5      |   20   |  100   |    560ms        |    o.k.
   10      |   10   |  100   |   1451ms        |    o.k.
   20      |    5   |  100   |   2719ms        |    o.k.



Test 2:

Method:
  getFolders()

Parameters:
  userID


Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |  100   |  100   |    713ms        |    o.k.
    5      |   20   |  100   |   1129ms        |    o.k.
   10      |   10   |  100   |   2159ms        |    o.k.
   20      |    5   |  100   |   4379ms        |    o.k.



Test 3:

Method:
  getName()

Parameters:
  folderID


Average Response Time:
```

concurrent requests	loops	total runs	average response-time	results o.k.?
1	100	100	399ms	o.k.
5	20	100	724ms	o.k.
10	10	100	1535ms	o.k.
20	5	100	3497ms	o.k.

Test 4:

Method:
  getDescription()

Parameters:
  folderID

Average Response Time:

concurrent requests	loops	total runs	average response-time	results o.k.?
1	100	100	400ms	o.k.
5	20	100	687ms	o.k.
10	10	100	1336ms	o.k.
20	5	100	2437ms	o.k.

Test 5:

Method:
  getMembers()

Parameters:
  folderID

Average Response Time:

concurrent requests	loops	total runs	average response-time	results o.k.?
1	100	100	455ms	o.k.
5	20	100	693ms	o.k.
10	10	100	1408ms	o.k.
20	5	100	2710ms	o.k.

Test 6:

```
Method:
  getQueries()

Parameters:
  folderID


Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |  100   |  100   |    404ms        |    o.k.
    5      |   20   |  100   |    707ms        |    o.k.
   10      |   10   |  100   |   1144ms        |    o.k.
   20      |    5   |  100   |   2582ms        |    o.k.



Test 7:

Method:
  getParents()

Parameters:
  folderID
  userID


Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |  100   |  100   |    449ms        |    o.k.
    5      |   20   |  100   |    723ms        |    o.k.
   10      |   10   |  100   |   1179ms        |    o.k.
   20      |    5   |  100   |   2633ms        |    o.k.



Test 8:

Method:
  getChildren()

Parameters:
  folderID


Average Response Time:

concurrent |        | total  | average         | results
```

requests	loops	runs	response-time	o.k.?
1	100	100	407ms	o.k.
5	20	100	645ms	o.k.
10	10	100	1191ms	o.k.
20	5	100	2310ms	o.k.

Test 9:

Method:
  getCollections()

Parameters:
  folderID

Average Response Time:

concurrent requests	loops	total runs	average response-time	results o.k.?
1	100	100	389ms	o.k.
5	20	100	811ms	o.k.
10	10	100	1274ms	o.k.
20	5	100	2201ms	o.k.

Test 9:

Method:
  getCommunity()

Parameters:
  folderID

Average Response Time:

concurrent requests	loops	total runs	average response-time	results o.k.?
1	100	100	408ms	o.k.
5	20	100	690ms	o.k.
10	10	100	1349ms	o.k.
20	5	100	2380ms	o.k.

Test 10:

Method:

```
  saveQuery()

Parameters:
  folderID
  userID
  query


Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-----------------------------------------------------------
    1      |  100   |  100   |    507ms        |    o.k.
    5      |   20   |  100   |    950ms        |    o.k.
   10      |   10   |  100   |   1792ms        |    o.k.
   20      |    5   |  100   |   3018ms        |    o.k.



Test 11:

Method:
  saveResults()

Parameters:
  folderID
  userID
  records


Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-----------------------------------------------------------
    1      |  100   |  100   |    502ms        |    o.k.
    5      |   20   |  100   |    908ms        |    o.k.
   10      |   10   |  100   |   1579ms        |    o.k.
   20      |    5   |  100   |   3076ms        |    o.k.



Test 12:

Method:
  saveRecommendedRecords()

Parameters:
  folderID
  recordIDs


Average Response Time:
```

concurrent requests	loops	total runs	average response-time	results o.k.?
1	100	100	745ms	o.k.
5	20	100	1267ms	o.k.
10	10	100	2198ms	o.k.
20	5	100	3887ms	o.k.

Test 13:

Method:
  saveRecommendedUsers()

Parameters:
  folderID
  UserIDs

Average Response Time:

concurrent requests	loops	total runs	average response-time	results o.k.?
1	100	100	421ms	o.k.
5	20	100	1267ms	o.k.
10	10	100	2198ms	o.k.
20	5	100	3887ms	o.k.

Test 14:

Method:
  saveRecommendedCommunites()

Parameters:
  folderID
  communityIDs

Average Response Time:

concurrent requests	loops	total runs	average response-time	results o.k.?
1	100	100	429ms	o.k.
5	20	100	647ms	o.k.
10	10	100	1390ms	o.k.
20	5	100	2436ms	o.k.

```
Test 15:

Method:
  saveRecommendedCollections()

Parameters:
  folderID
  collectionIDs


Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |  100   |  100   |    430ms        |    o.k.
    5      |   20   |  100   |    588ms        |    o.k.
   10      |   10   |  100   |   1281ms        |    o.k.
   20      |    5   |  100   |   2352ms        |    o.k.



Test 16:

Method:
  addModifyCollection()

Parameters:
  collectionID
  collectionName


Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |  100   |  100   |    391ms        |    o.k.
    5      |   20   |  100   |    659ms        |    o.k.
   10      |   10   |  100   |   1099ms        |    o.k.
   20      |    5   |  100   |   2042ms        |    o.k.



Test 17:

Method:
  updatePersonalCollections()

Parameters:
  userID
  collectionIDs
```

```
Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-----------------------------------------------------------
    1      |  100   |  100   |    411ms        |    o.k.
    5      |   20   |  100   |    664ms        |    o.k.
   10      |   10   |  100   |   1076ms        |    o.k.
   20      |    5   |  100   |   2197ms        |    o.k.
```

3.2.3.3 Test results and summary

The response times of all methods are below one second, typically around half a second if only one call is made at once. The response times increase moderately on heavier load. No errors occurred during the tests.

3.2.4 Filtering and Recommendation Service

- **Testers:** M.Elena Renda
  Istituto di Scienza e Tecnologie della Informazione – CNR, Pisa
  renda@iei.pi.cnr.it
- **Test date: April 23rd, 2003**
- **Scope of test:**
  - *FilteredSearch*
  - *getNewRecords*
  - *updateFolderProfile*
  - *createFolder*
  - *addRecord*
  - *setRecommendationYesNo*
  - *addRating*
  - *deleteRecord*
  - *destroyFolder*
  - *deleteUser*
- **Test environment:** Apache JMeter 1.8.1, Mozilla 1.3 [Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.3) Gecko/20030312] on Windows 2000

3.2.4.1 Test plan

For each API of the Filtering and Recommendation Service a test plan was created with the Apache JMeter test kit, simulating an XML-RPC request to the system. For these test suites it was determined how fast the system responded to each request. After that for each API call the system was subjected to a critical evaluation of its performance under high load, to determine its scalability. Therefore the test suits were started concurrently with an increasing number of simultaneous runs. Again the response times were determined. We report the min, the max and the average response time, and the error percentage of all tests. It was also checked how reliable the system was under the high load, by verifying the results and noting errors.

3.2.4.2 Test log

In the tables that follow are reported the results of all API tests, in terms of response time and error percentage.

Method: FilteredSearch
Parameters: query
maxRecordNo
folderID
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	237	50	1622	0.00%	4.1/sec
5	40	200	137	50	941	0.00%	14.6/sec
10	20	200	1837	60	6419	0.00%	3.3/sec
15	15	225	3431	80	7281	0.00%	3.5/sec
20	10	200	3627	60	10214	0.00%	3.6/sec

Method: getNewRecords
Parameters: query
maxRecordNo
folderID
userID
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	235	60	1563	0.00%	4.0/sec
5	40	200	130	60	1993	0.00%	15.6/sec
10	20	200	2065	90	6038	0.00%	3.2/sec
15	15	225	3470	120	7851	0.00%	3.5/sec
20	10	200	3813	80	10295	0.00%	3.6/sec

Method: updateFolderProfile
Parameters: folderID
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	3254	451	7070	0.00%	18.4/min
5	40	200	3121	430	6669	0.00%	1.5/sec
10	20	200	3014	491	8382	0.00%	2.8/sec
15	15	225	4370	430	10996	0.00%	2.9/sec
20	10	200	5181	470	13389	0.00%	2.9/sec

Method: createFolder
Parameters: folderID
userID
recommendationValue
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	190	50	1492	0.00%	4.9/sec
5	40	200	52	20	201	0.00%	19.9/sec
10	20	200	1603	40	5278	0.00%	3.4/sec
15	15	225	2855	50	9093	0.00%	3.6/sec
20	10	200	3697	50	9634	0.00%	3.6/sec

Method: addRecord
Parameters: recordID
folderID
userID
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	181	30	1172	0.00%	5.2/sec
5	40	200	45	10	551	0.00%	20.2/sec
10	20	200	2096	50	4726	0.00%	3.4/sec
15	15	225	3134	121	7261	0.00%	3.6/sec
20	10	200	3996	10	9634	0.00%	3.6/sec

Method: setRecommendationYesNo
Parameters: folderID
value
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	180	0	1522	0.00%	5.2/sec
5	40	200	5	0	290	0.00%	20.9/sec
10	20	200	1951	130	4626	0.00%	3.9/sec
15	15	225	3003	50	8643	0.00%	3.6/sec
20	10	200	4065	251	9184	0.00%	3.8/sec

Method: addRating
Parameters: recordID
folderID
userID
ratingValue
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	192	0	1252	0.00%	4.8/sec
5	40	200	33	0	761	0.00%	21.2/sec
10	20	200	1788	50	4636	0.00%	3.4/sec
15	15	225	2992	100	7892	0.00%	3.7/sec
20	10	200	3786	60	10485	0.00%	3.7/sec

Method: deleteRecord
Parameters: recordID
folderID
userID
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	13	0	50	0.00%	83.3/sec
5	40	200	81	0	852	0.00%	35.4/sec
10	20	200	188	0	2073	0.00%	33.1/sec
15	15	225	343	0	3355	0.00%	30.3/sec
20	10	200	392	0	2724	0.00%	29.8/sec

Method: destroyFolder
Parameters: folderID
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	33	0	100	0.00%	26.5/sec
5	40	200	104	0	741	0.00%	34.8/sec
10	20	200	227	10	2394	0.00%	28.9/sec
15	15	225	237	10	2463	0.00%	35.4/sec
20	10	200	326	10	3175	0.00%	30.1/sec

Method: deleteUser
Parameters: userID
Results:

concurrent

calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	200	29	0	61	0.00%	30.3/sec
5	40	200	107	0	871	0.00%	33.5/sec
10	20	200	215	0	2283	0.00%	31.5/sec
15	15	225	324	0	4336	0.00%	30.0/sec
20	10	200	284	0	2954	0.00%	29.6/sec

3.2.4.3 Test results and summary

Generally, the system showed efficiency in handling all the tested APIs by responding to all the requests within short time. The system responded to all API calls without errors.

3.2.5 Mediator Service

- **Tester:** Sascha Kriewel, University of Duisburg-Essen (sascha.kriewel@uni-duisburg.de)
- **Test date:** May 5th, 2003
- **Scope of test:**
  - *getService*
  - *getServiceDescription*
  - *getErrorLog*
  - *addUser*
  - *deleteUser*
  - *getUserIds*
  - *getUserInfo*
- **Test environment:** JMeter 1.8.1, Mozilla 1.2b [Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.2b) Gecko/20021016] on Debian Linux (Kernel 2.4)

3.2.5.1 Test plan

To measure the API efficiency, each method call was simulated using the XML-RPC with the help of the JMeter application. For each method, a test plan was made that is composed of an XML document containing the method name and the parameters passed, and a thread group containing the information of concurrent calls sent and number of times these cocurrent calls are sent. The following set of concurrent calls are sent to each method: 1, 5, 10, 15, 20 and each of these concurrent calls are sent several times to determine an average result. During this, method response time and errors are observerved.

3.2.5.2 Test log

```
Test 1
------

Method:      getService

Parameters:  serviceType = ME

Results:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |  100   |  100   |   928 ms ~ 1s   |    x
    5      |   20   |  100   |  1378 ms ~ 1s   |    x
   10      |   10   |  100   |  1433 ms ~ 1s   |    x
   15      |    7   |  105   |  1536 ms ~ 2s   |    x
   20      |    5   |  100   |  2062 ms ~ 2s   |    x


Test 2
------

Method:       getServiceDescription

Parameters:   serviceID = CW665


concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |  100   |  100   |   907ms ~ 1s    |    x
    5      |   20   |  100   |  1623ms ~ 2s    |    x
   10      |   10   |  100   |  1841ms ~ 2s    |    x
   15      |    7   |  105   |  1981ms ~ 2s    |    x
   20      |    5   |  100   |  2329ms ~ 2s    |    x


Test 3
------

Method:       getErrorLog
```

```
Parameters:   serviceID = CW665

Results:
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-----------------------------------------------------------
    1      |  100   |  100   |  1273 ms ~ 1s   |    x
    5      |   20   |  100   |  1970 ms ~ 2s   |    x
   10      |   10   |  100   |  2319 ms ~ 2s   |    x
   15      |    7   |  105   |  1887 ms ~ 2s   |    x
   20      |    5   |  100   |  1921 ms ~ 2s   |    x


Test 4
------

Method:       addUser

Parameters:   id       = CW665_97654
              username = testuser
              password = testpass
              mailAddr = test@email.test
              folderId = CW665_98765

Results:
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |  100   |  100   |   809 ms ~ 1s   |    x
    5      |   20   |  100   |  1111 ms ~ 1s   |    x
   10      |   10   |  100   |  1121 ms ~ 1s   |    x
   15      |    7   |  105   |  1269 ms ~ 1s   |    x
   20      |    5   |  100   |  1347 ms ~ 1s   |    x


Test 5
------

Method:       deleteUser

Parameters:   id = CW665_97654

Results

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |  100   |  100   |   570 ms ~ 1s   |    x
    5      |   20   |  100   |  1318 ms ~ 1s   |    x
   10      |   10   |  100   |  1067 ms ~ 1s   |    x
   15      |    7   |  105   |  1379 ms ~ 1s   |    x
   20      |    5   |  100   |  1274 ms ~ 1s   |    x
```

```
Test 6
------

Method:       getUserIds

Parameters:   none

Results:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |  100   |  100   |  1124 ms ~1s    |    x
    5      |   20   |  100   |  1357 ms ~1s    |    x
   10      |   10   |  100   |  1401 ms ~1s    |    x
   15      |    7   |  105   |  1873 ms ~2s    |    x
   20      |    5   |  100   |  1940 ms ~2s    |    x


Test 7
------

Method:       getUserInfo

Parameters:   userIds = [ CW665_20164 ]


Results:
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |  100   |  100   |    843 ms ~1s   |    x
    5      |   20   |  100   |   1074 ms ~1s   |    x
   10      |   10   |  100   |   1356 ms ~1s   |    x
   15      |    7   |  105   |   1361 ms ~1s   |    x
   20      |    5   |  100   |   1632 ms ~2s   |    x
```

3.2.5.3 Test results and summary

All the methods are working efficiently even under simulated heavy load. No errors occured during performance testing, and the average response times didn't exceed 2 seconds even with numerous concurrent merhod calls.

3.2.6 Rating Management Service

- **Tester:** Wido Wirsam, Fraunhofer FIT (wido.wirsam@fit.fraunhofer.de)
- **Test date:** May 5th, 2003
- **Scope of test:**
  - *saveRating*
  - *getFolderRatings*
  - *getRecordRatings*

– *getUserRatings*

- **Test environment:** JMeter 1.8.1, on Windows XP

3.2.6.1 Test plan

To measure the API efficiency, each method call was simulated using the XML-RPC interface. The simultanous execution was performed using the JMeter application. For each method, a test plan has been created with the methods name and parameters encoded in XML. Each method was called with the following number of concurrent threads: 1, 5, 10, 20. Those calls have been repeated until the total number of 100 calls of every method was reached to determine an average result. During this, method response times and errors were observed.

3.2.6.2 Test log

```
Test 1:

Method:
  saveRating()

Parameters:
  recordID, folderID, userID, ratingValue


Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-----------------------------------------------------------
    1      |  100   |  100   |    227ms        |    o.k.
    5      |   20   |  100   |    363ms        |    o.k.
   10      |   10   |  100   |    818ms        |    o.k.
   20      |    5   |  100   |   1216ms        |    o.k.



Test 2:

Method:
  getFolderRatings()

Parameters:
  folderID, timeStamp


Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-----------------------------------------------------------
    1      |  100   |  100   |    237ms        |    o.k.
    5      |   20   |  100   |    449ms        |    o.k.
   10      |   10   |  100   |    763ms        |    o.k.
   20      |    5   |  100   |   1604ms        |    o.k.
```

```
Test 3:

Method:
  getRecordRatings()

Parameters:
  recordID, timeStamp


Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-----------------------------------------------------------
    1      |  100   |  100   |    241ms        |    o.k.
    5      |   20   |  100   |    399ms        |    o.k.
   10      |   10   |  100   |    817ms        |    o.k.
   20      |    5   |  100   |   1229ms        |    o.k.



Test 4:

Method:
  getUserRatings()

Parameters:
  userID, timeStamp


Average Response Time:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-----------------------------------------------------------
    1      |  100   |  100   |    243ms        |    o.k.
    5      |   20   |  100   |    428ms        |    o.k.
   10      |   10   |  100   |    923ms        |    o.k.
   20      |    5   |  100   |   1359ms        |    o.k.
```

3.2.6.3 Test results and summary

The response times of all methods are below 250 milliseconds, if only one call is made at once. The response times increase moderately on heavier load. No errors occurred during the tests.

3.2.7 Search and Browse Service

- **Tester:** Saadia Malik, University of Duisburg-Essen (malik@is.informatik.uni-duisburg.de)
- **Test date:** April 8th, 2003

- **Scope of test:**
  - *initiateSearch*
  - *search*
  - *saveResults*
  - *saveQuery*
  - *getFolderQueries*
  - *getCollections*
  - *getPersonalCollections*
  - *getFolderCollections*
  - *getRecords*
  - *getQuery*
  - *getQueryHistory*
  - *getSchemas*
  - *getNewForFolder*
  - *getResultHistory*
  - *getAttributeValues*
  - *filteredSearch*
- **Test environment:** JMeter 1.8.1 on Debian Linux (Kernel 2.4)

3.2.7.1 Test plan

API efficiency is measured by simulating each method call with XML-RPC. For each API, a test plan is made using JMeter. A test plan is composed of an XML document containing the name of the methods and describing the parameters passed, a declaration of concurrent calls and the number of times these concurrent calls are sent to the method. An average result is calculated for the following set of concurrent calls: 1, 5, 10, 15, 20 and each test run repeated several times. In the meanwhile, average response time and errors are observed.

3.2.7.2 Test log

```
Test 1
------

Method:  initiateSearch

Parameters:  id=id14557,
             folderid=CW665_21300,
             userId=CW665_20247

Results:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-----------------------------------------------------------
    1      |   20   |   20   |    13 ms        |    x
    5      |   20   |  100   |   102 ms        |    x
   10      |   20   |  200   |   370 ms        |    x
```

```
   15      |   20   |  300   |   478 ms        |    x
   20      |   20   |  400   |   556 ms        |    x


Test 2
------

Method:  search

Parameters:  sessionId=id14557
             query=

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sbquery SYSTEM
"http://woodstock.is.informatik.uni-duisburg.de:15210/ac/query.dtd">
<sbquery schema="dc">
  <condition weight="+" field="subject">
    <field-condition predicate="$cw$" value="system"/>
  </condition>
  <collection id="CO_all"/>
</sbquery>


concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |   20   |  20    |    6376 ms ~6s  |    85%
    2      |   20   |  40    |    4501 ms ~5s  |    0%

With more than two concurrent requests the method started to behave
very unstable, making it basically untestable. It is currently being
reimplemented to take care of this.

Problems:
- java.lang.OutOfMemoryError


Test 3
------

Method:  getSchemas

Parameters:  sessionId=id14557

Results:
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |   20   |   20   |   154 ms        |    x
    5      |   20   |  100   |   495 ms        |    x
   10      |   20   |  200   |   872 ms        |    x
   15      |   20   |  300   |  1322 ms        |    x
   20      |   20   |  400   |  1820 ms        |    x
```

```
Test 4
------

Method:  getCollections

Parameters:  sessionId=id14557

Results:
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |   20   |   20   |   166 ms        |    x
    5      |   20   |  100   |   545 ms        |    x
   10      |   20   |  200   |   918 ms ~ 1 s  |    x
   15      |   20   |  300   |  1470 ms ~ 1 s  |    x
   20      |   20   |  400   |  2463 ms ~ 2 s  |    x


Test 5
------

Method:  getPersonalCollections

Parameters:  sessionId=id14557

Results

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |   20   |   20   |   191 ms        |    x
    5      |   20   |  100   |   465 ms        |    x
   10      |   20   |  200   |   901 ms ~ 1 s  |    x
   15      |   20   |  300   |  1278 ms ~ 1 s  |    x
   20      |   20   |  400   |  1779 ms ~ 2 s  |    x


Test 6
------

Method:  getFolderCollections

Parameters:  sessionId=id14557

Results:

concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |   20   |   20   |   151 ms        |    x
    5      |   20   |  100   |   436 ms        |    x
   10      |   20   |  200   |   839 ms ~ 1s   |    x
   15      |   20   |  300   |  1112 ms ~ 1s   |    x
```

```
   20      |   20   |  400   |  1886 ms ~ 2s   |    x


Test 7
------

Method:  getFolderQueries

Parameters:  sessionId=id14557


Results:
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |   20   |   20   |   3105 ms ~ 3s  |    x
    5      |   20   |  100   |   3071 ms ~ 3s  |    x
   10      |   20   |  200   |   2894 ms ~ 3s  |    x
   15      |   20   |  300   |   2640 ms ~ 3s  |    x
   20      |   20   |  400   |   2732 ms ~ 3s  |    x


Test 8
------

Method:  getQuery

Parameters:  sessionId=id14557
             queryId=SB438_id14557_1049729936633_2


Results:
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |   20   |   20   |   21 ms         |    x
    5      |   20   |  100   |  100 ms         |    x
   10      |   20   |  200   |  381 ms         |    x
   15      |   20   |  300   |  517 ms         |    x
   20      |   20   |  400   |  680 ms         |    x


Test 9
------

Method:  getQueryHistory

Parameters:  sessionId=id14557

Results:
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |   20   |   20   |     43 ms       |    x
```

```
    5      |   20   |  100   |     80 ms       |    x
   10      |   20   |  200   |    393 ms       |    x
   15      |   20   |  300   |    489 ms       |    x
   20      |   20   |  400   |    508 ms       |    x


Test 10
-------

Method:  getRecord

Parameters:  sessionId=id14557
             recordId=
             AC143_oai_dc_oai:CompSciPreprints:Conferences/0202171

Results
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |   20   |   20   |   151 ms        |    x
    5      |   20   |  100   |   810 ms ~ 1s   |    x
   10      |   20   |  200   |  1305 ms ~ 1s   |    x
   15      |   20   |  300   |  1761 ms ~ 2s   |    x
   20      |   20   |  400   |  2189 ms ~ 2s   |    x


Test 11
-------

Method:  getResultHistory

Parameters:  sessionId=id14557


Results:
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |   20   |   20   |    80 ms        |    x
    5      |   20   |  100   |   154 ms        |    x
   10      |   20   |  200   |   311 ms        |    x
   15      |   20   |  300   |   566 ms        |    x
   20      |   20   |  400   |   799 ms        |    x


Test 12
-------

Method:  saveResults

Parameters: sessionId=id14557
            list of recordIds=
            AC143_oai_dc_oai:CompSciPreprints:Conferences/0202171
```

```
Results:

concurrent |        | total  | average          | results
requests   | loops  | runs   | response-time    | o.k.?
-------------------------------------------- ------------
    1      |   20   |   20   |   2657 ms        |    x
    5      |   20   |  100   |   3220 ms ~ 3s   |    x
   10      |   20   |  200   |   4203 ms ~ 4s   |    x
   15      |   20   |  300   |   4498 ms ~ 4s   |    x

Because testing tried to write to the same BSCW object several times
at once, the CWS threw some errors during the concurrency tests. These
errors cannot occur during normal use of the system, but are specific
to the testing method used, so they weren't further noted.


Test 13
-------

Method:  saveQuery

Parameters: sessionId=id14558
            query=

<struct>
  <member>
     <name>id</name>
     <value>SBtest_query1</value>
  </member>
  <member>
     <name>id</name>
     <value>test query 1</value>
  </member>
  <member>
     <name>queryString</name>
     <value>&lt;?xml version="1.0" encoding="UTF-8"?&gt;
&lt;!DOCTYPE sbquery SYSTEM
"http://austin.is.informatik.uni-duisburg.de:15210/ac/query.dtd"&gt;
&lt;sbquery schema="dc"&gt; &lt;condition weight="+" field="subject"&gt;
&lt;field-condition predicate="$cw$" value="system"/&gt; &lt;/condition&gt;
&lt;collection id="CO_all"/&gt; &lt;/sbquery&gt;</value>
  </member>
</struct>


Results:
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |   20   |   20   |   3388 ms       |    x
    5      |   20   |  100   |   2904 ms ~ 3   |    x
   10      |   20   |  200   |   2876 ms ~ 3   |    x
   15      |   20   |  300   |   3253 ms ~ 3   |    x
```

```
see remarks for test 13 above


Test 14
-------

Method:  getNewForFolder

Parameters:  sessionId


Results:
concurrent |        | total  | average             | results
requests   | loops  | runs   | response-time       | o.k.?
-----------------------------------------------------------
    1      |   25   |   25   |    4646 ms ~  4 s   |  x
    5      |    5   |   25   |   17032 ms ~ 17 s   |  x


Test 15.1
---------

Method:  getAttributeValues

Parameters:  sessionId=id14558
             schemaName=oai_dc
             attributeName="title"
             maxNo=20

Results:
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-------------------------------------------------------------
    1      |  200   |  200   |    154 ms       |    x
    5      |   40   |  200   |    670 ms ~1s   |    x
   10      |   20   |  200   |   1135 ms ~1s   |    x
   15      |   15   |  225   |   1577 ms ~2s   |    x
   20      |   10   |  200   |   1897 ms ~2s   |    x


Test 15.2
---------

Method:  getAttributeValues

Parameters:  sessionId=id14558
             schemaName=oai_dc
             attributeName="title"
             maxNo=1000

Description: high data volume

Results:
```

```
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-----------------------------------------------------------
    1      |  200   |  200   |    398 ms       |    x
   10      |   20   |  200   |   2844 ms ~3s   |    x
   20      |   10   |  200   |   5178 ms ~5s   |    x


Test 15.3
---------

Method:  getAttributeValues

Parameters:  sessionId
             schemaName
             attributeName="description"
             maxNo=5000

Description: very high data volume

Results:
concurrent |        | total  | average         | results
requests   | loops  | runs   | response-time   | o.k.?
-----------------------------------------------------------
    1      |  200   |  200   |   2733 ms ~3s   |    x



Test 16
-------

Method: filteredSearch

Parameters:  sessionId
             query

Results:
concurrent |        | total  |  average            | results
requests   | loops  | runs   | response-time       | o.k.?
-----------------------------------------------------------
    1      |  20    |   20   |    19836 ms ~  20 s |  -

the same problems as for the method search() also occured during
tests for this method, see test 2 above
```

3.2.7.3 Test results and summary

All the methods are working efficiently with the exceptions of *search* and *filteredSearch*. Under normal conditions, average response time was 2 seconds and under heavy load, response time didn't exceed 4 seconds. In case of *search* and *filteredSeach*, processing took very long, got stuck and sometime ended up with an 'out of memory' exception. *getNewFolder* either throws exception or takes a very long time to produce any result.

3.3 GUI

3.3.1 Access Service

- **Tester:** Sascha Kriewel, University of Duisburg-Essen (sascha.kriewel@uni-duisburg.de)
- **Test date:** April 7th, 2003
- **Scope of test:**
  - Registering a new archive
  - Editing archive information
  - Deleting an archive
- **Test environment:** JMeter 1.8.1, Mozilla 1.2b [Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.2b) Gecko/20021016] on Debian Linux (Kernel 2.4)

3.3.1.1 Test plan

For each use case a test plan was created with the Apache JMeter test kit. These test runs were composed of a number of HTTP requests that simulate the use of the system by a human user. Each test suite began with initiating a new Archive Management session, and then ran through the actions that are necessary to complete the use case as described previously during the functionality test (section 2.2).

For these test suites it was determined how fast the system responded to each user request that is part of the use case, and how long it took to complete the total use case. Each test suite was run at least 100 times, to get average results for the response times. After that for each use case the system was subjected to a critical evaluation of its performance under high load, to determine its scalability. Therefore the test suits were started concurrently with an increasing number of simultaneous runs. Again the response times were determined. It was also checked how reliable the system was under the high load, by verifying the results and noting errors.

3.3.1.2 Test log

```
Use Case: Register Archive

Description of Test:

- Call AS GUI from user folder
- start registration
- register URL
- complete registration

Results:

concurrent calls     |    1   |    5   |   10   |   15   |   20
loops                |  200   |   40   |   20   |   15   |   10
total calls          |  200   |  200   |  200   |  225   |  200
-------------------------------------------------------------------
TASK:                |        |        |        |        |
- Call AS GUI        |  861ms | 1613ms | 2046ms | 2809ms | 3539ms
- Start Registration |  359ms |  801ms |  776ms | 1047ms | 1304ms
- Register URL       | 2379ms | 3625ms | 3977ms | 5218ms | 5817ms
```

- Complete Registr.	356ms	888ms	973ms	1320ms	1820ms
Total Use Case:	~4s	~7s	~8s	~10s	~13s
all responses o.k.	x	x	x	x	x

Use Case: Edit Registration Information

Description of Test:

- Call AS GUI from user folder
- select archive and choose to edit
- send new information

Results:

concurrent calls	1	5	10	15	20
loops	200	40	20	15	10
total calls	200	200	200	225	200
TASK:					
- Call AS GUI	603ms	1520ms	2740ms	3931ms	4855ms
- Edit Information	204ms	554ms	996ms	1362ms	1823ms
- Save Information	240ms	623ms	1118ms	1476ms	1877ms
Total Use Case:	~1s	~3s	~5s	~7s	~9s
all responses o.k.	x	x	x	x	x

Use Case: Delete Archive

Description of Test:

- Call AS GUI from user folder
- delete selected archive

Results:

concurrent calls	1	5	10	15	20
loops	200	40	20	15	10
total calls	200	200	200	225	200
TASK:					
- Call AS GUI	599ms	819ms	1059ms	1856ms	2798ms
- Delete Archive	531ms	2052ms	3828ms	4118ms	4223ms
Total Use Case:	~1s	~3s	~5s	~6s	~7s
all responses o.k.	x	x	x	x	x

3.3.1.3 Test summary

The system performed efficiently for all use cases associated with Archive Management. With the exception of URL registration, the average response time for each single task within the use cases didn't exceed one second under controlled conditions. URL registration is an obvious exception since it needs to issue its own HTTP request to an outside archive and is therefore subject to network latency. All use cases could be completed in under four seconds.

Under increasingly heavier load, the system scaled well and showed no errors.

All the use cases could be completed with other services missing. The Access Service GUI can be called independently from the CYCLADES prototype, and archives can still be registered, edited and deleted. Without a Collection Service running, these won't result in immediate changes in the system's collection.

3.3.2 Collection Service

- **Testers:** Leonardo Candela, M.Elena Renda

  Istituto di Scienza e Tecnologie della Informazione – CNR, Pisa

  {candela, renda}@iei.pi.cnr.it

- **Test date: April 17th, 2003**

- **Scope of test:**

  - Creating a new collection
  - Editing a collection description
  - Deleting a collection

- **Test environment:** Apache JMeter 1.8.1, Mozilla 1.3 [Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.3) Gecko/20030312] on Windows 2000

3.3.2.1 Test plan

For each Use Case a test plan was created with the Apache JMeter test kit. These test runs were composed of a number of HTTP requests that simulate the use of the system by a human user. Each test suite began with opening the Collection Management Window, and then ran through the actions that are necessary to complete the use case as described previously during the functionality test.

For these test suites it was determined how fast the system responded to each user request that is part of the use case (therefore it could be determined how long it took to complete the total use case). After that for each use case the system was subjected to a critical evaluation of its performance under high load, to determine its scalability. Therefore the test suits were started concurrently with an increasing number of simultaneous runs. We report the min, the max and the average response time, and the error percentage of all tests. It was also checked how reliable the system was under the high load, by verifying the results.

3.3.2.2 Test log

Use Case:	CREATE COLLECTION
Actions:	CS home page
	newCollectionForm
	SUBMIT
Results:	

concurrent calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	600	2490	631	5808	0,00%	28,0/min
5	40	600	1815	341	4887	0,00%	17,0/min
10	20	600	2136	371	6480	0,00%	31,6/min
15	15	675	2529	330	11075	0,00%	47,4/min
20	10	600	10905	861	31024	0,00%	55,3/min

Use Case: EDIT COLLECTION
Actions: CS home page
Select collection
Edit collection form
SUBMIT
Results:

concurrent calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	800	1279	751	2644	0,00%	1,6/sec
5	40	800	1326	851	4376	0,00%	23,5/min
10	20	800	1336	651	11086	0,00%	39,9/min
15	15	900	1303	70	4687	0,00%	57,5/min
20	10	800	8019	1182	14521	0,00%	1,3/sec

Use Case: DELETE COLLECTION
Actions: CS home page
Select collection
show collection data
SUBMIT
Results:

concurrent calls	loops	total	Avg ms	Min ms	Max ms	Error%	Rate
1	200	800	1517	781	3455	0,00%	46,5/min
5	40	800	989	721	2263	0,00%	23,5/min
10	20	800	1268	691	5188	0,00%	43,3/min
15	15	900	3351	761	8292	0,00%	1,5/sec
20	10	800	7958	981	16554	0,00%	1,3/sec

3.3.2.3 Test results and summary

Generally, the system showed efficiency in handling most of the tested use cases by responding to user input and requests within short time.

The Collection Service is not able to create a new collection without a working Collaborative Work Service, because it cannot notify the new collection to this service; further, if the collection is related to a given archive, it has to get the information of the archive from the Access Service.

Without the Mediator Service the Collection Service has no information about other working Cyclades services, and no user information, so that it cannot display the personal collection of the user.

3.3.3 Collaborative Work Service

- **Tester:** Wido Wirsam, Fraunhofer FIT (wido.wirsam@fraunhofer.fit.de)
- **Test date:** April 29th, 2003

- **Scope of test:**
  - copy one record from a community folder to a private folder
  - invite a user to a community
  - rating a record
- **Test environment:** JMeter 1.8.1, on Windows XP

3.3.3.1 Test plan

For each Use Case a test plan was created with the Apache JMeter test kit. These test runs were composed of a number of HTTP requests that simulate the use of the system by a human user. Each test suite began with initiating a new Archive Management session, and then ran through the actions that are necessary to complete the use case as described previously during the functionality test.

For these test suites it was determined how fast the system responded to each user request that is part of the use case, and how long it took to complete the total use case. Each test suite was run at least 100 times, to get average results for the response times. After that for each use case the system was subjected to a critical evaluation of its performance under high load, to determine its scalability. Therefore the test suits were started concurrently with an increasing number of simultaneous runs. Again the response times were determined. It was also checked how reliable the system was under the high load, by verifying the results and noting errors.

3.3.3.2 Test log

```
Use Case: Copy a record from a community folder to a private folder

Description of Test:
- call CWS-UI and authorize
- navigate to community 'Physics Gravity'
- copy Record 'Nonlinear earthquake response of concrete gravity dam systems'
- navigate to home folder
- navigate to private folder
- paste record


Results:

concurrent calls             |     1  |    5   |   10   |   20   |
loops                        |   100  |   20   |   10   |    5   |
total calls                  |   100  |  100   |  100   |  100   |
-------------------------------------------------------------------
TASK:                        |
- call CWS-UI                | 4202ms | 3930ms | 4791ms | 9269ms |
- navigate to community      |  990ms | 1176ms | 2521ms | 6501ms |
- copy Record                | 4406ms | 4894ms | 6276ms |10894ms |
- navigate to home folder    | 3086ms | 3186ms | 4894ms | 9301ms |
- navigate to private folder | 4920ms | 1286ms | 2236ms | 3963ms |
- paste record               | 8521ms | 4340ms | 4683ms | 4638ms |
-------------------------------------------------------------------
Total Use Case               |21125ms |18812ms |25401ms |44566ms |
all responses o.k.           |  o.k.  |  o.k.  |  o.k.  |  o.k.  |
```

Use Case: invite a Member to a community

Description of Test:
- navigate to home folder
- navigate to community 'Physics - Mechanics'
- choose 'invite Member' from menu
- select a member to invite
- be redirected to community again

Results:

concurrent calls	1	5	10	20
loops	100	20	10	5
total calls	100	100	100	100
TASK:				
- home Folder	3749ms	3513ms	4404ms	7595ms
- navigate to community	5062ms	5154ms	7217ms	14542ms
- call 'invite Member'	3329ms	3048ms	3151ms	4501ms
- select member	5489ms	6464ms	8953ms	18015ms
- call home folder	2183ms	2643ms	5440ms	13630ms
Total Use Case	19812ms	20822ms	29165ms	58283ms
all responses o.k.	o.k.	o.k.	o.k.	o.k.

Use Case: rating a record in a private folder

Description of Test:
- navigate to home folder
- navigate to private folder 'private'
- navigate to private folder 'ratingFolder'
- choose 'Rate' from contextmenu of record 'On Morality and Chemistry'
- choose 'good' as rating value
- be redirected to folder again

Results:

concurrent calls	1	5	10	20
loops	100	20	10	5
total calls	100	100	100	100
TASK:				
- home Folder	1501ms	3678ms	4073ms	7289ms
- navigate to 'private'	3613ms	3327ms	3505ms	5667ms
- navigate to 'ratingFolder'	3615ms	3373ms	3665ms	5293ms
- choose 'Rate' from menu	3119ms	2717ms	2407ms	3607ms
- choose 'good' as value	4125ms	4330ms	5449ms	8138ms
- be redirected to folder	1100ms	915ms	1756ms	4051ms

```
-------------------------------------------------------------------
Total Use Case                |17037ms |18340ms |20855ms |34045ms |
all responses o.k.            |  o.k.  |  o.k.  |  o.k   |  o.k   |
```

3.3.3.3 Test summary

The system performed efficiently for all use cases associated with Collaborative Work Service.

Under increasingly heavier load the system scaled well and showed no errors.

Without a Mediator Service, the GUI of the CWS was still directly accessible with almost full functionality. It was not possible, however, to invite users to communities or projects via e-mail address, i.e. to invite persons that were no registered CYCLADES users.

Without a working Collection Service, it was not possible to associate collections to folders. Once a list of collections had been established, a breakdown of the Collection Service merely kept the CWS from updating this list of collections.

Without a Search and Browse Service, it was not possible to start search and browse sessions from the CWS.

Without a Filtering and Recommendation Service, it was not possible to receive recommendations for folders. Without an Access Service, received record recommendations could not be retrieved, because recommendations are made by record identifier.

3.3.4 Mediator Service

- **Tester:** Sascha Kriewel, University of Duisburg-Essen (sascha.kriewel@uni-duisburg.de)
- **Test date:** May 2nd, 2003
- **Scope of test:**
  - Registering a new user
  - Logging in to the system
  - Changing user details
  - Un-registering from the system
- **Test environment:** JMeter 1.8.1, Mozilla 1.2b [Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.2b) Gecko/20021016] on Debian Linux (Kernel 2.4)

3.3.4.1 Test plan

For the above use cases a single test plan was created with the Apache JMeter test kit. This test run would first register a new user with a unique username and email address, then login as that user, change the password, then use the password to un-register from the system. This test run was composed of a number of HTTP requests that simulate the use of the system by a human user.

For this test suite it was determined how fast the system responded to each user request that is part of a use case, and how long it took to complete the total use cases. The test suite was run 100 times to get an average result for the response times. After that the system was subjected to a critical evaluation of its performance under high load, to determine its scalability. For this purpose the test plan was modified to work with first 2, then 5 concurrent test runs. Each test run used a unique user name and email address, so as not to produce errors. Again average response times were determined by running the modified test suites several times. It was also checked how

reliable the system was under the simulated conditions, by verifying the results in a browser, and noting any errors.

3.3.4.2 Test log

```
Complete Test Run:

Use Cases:
  1. Register as a new user
  2. Login to the system
  3. Change registration details (password)
  4. Un-Register from the system

Description of Test:

Use Case 1:
- open the login page
- open the registration form
- enter registration information and send

Use Case 2:
- open the login page
- enter username and password, and send

Use Case 3:
- open options menu
- select change password dialog
- enter old and new password, and send

Use Case 4:
- open options menu
- select remove user dialog
- enter password and send

Results:

concurrent calls     |    1   |    2   |    5
loops                |  100   |   50   |   20
total calls          |  100   |  100   |  100
-----------------------------------------------
TASK:                |        |        |
- Open Login Form    |  173ms |  316ms |  303ms
- Open Regsitration  |  155ms |  209ms |  206ms
- Register New User  | 1141ms | 1385ms | 1312ms
-----------------------------------------------
Total Use Case 1:    |   ~1s  |   ~2s  |   ~2s
-----------------------------------------------
- Open Login Form    |  159ms |  191ms |  222ms
- Login To System    |  354ms |  405ms |  644ms
-----------------------------------------------
Total Use Case 2:    |   <1s  |   <1s  |   ~1s
-----------------------------------------------
- Open Options Menu  |  159ms |  202ms |  199ms
- Open Passwd Dialog |  155ms |  195ms |  239ms
```

- Change Password	985ms	1314ms	1683ms
Total Use Case 3:	~1s	~2s	~2s
- Open Options Menu	158ms	201ms	200ms
- Open Delete Dialog	160ms	185ms	763ms
- Un-Register User	4081ms	5284ms	6180ms
Total Use Case 4:	~4s	~6s	~7s
all responses o.k.	x	x	x

3.3.4.3 Test summary

The system handled all mediator use cases efficiently, and scaled well. Registering as a new user and logging in to the system was quick and worked without a great delay. Even with several users trying to register, or log in, at the same time, no severe problems became evident.

The use cases of the Mediator Service do not work without a properly running Collaborative Work Service, as this service is necessary to store and retrieve user details like username and password. It will work without any of the other services, without problems.

3.3.5 Search and Browse Service

- **Tester:** Sascha Kriewel, University of Duisburg-Essen (sascha.kriewel@uni-duisburg.de)
- **Test date:** April 7th, 2003
- **Scope of test:**
  - Browsing system collections
  - Browsing folder collections
  - Browsing personal collections
  - Requesting new records for folder
  - Saving results
  - Saving queries
  - Submitting query without personalization
- **Test environment:** JMeter 1.8.1, Mozilla 1.2b [Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.2b) Gecko/20021016] on Debian Linux (Kernel 2.4)

3.3.5.1 Test plan

For each use case a test plan was created with the Apache JMeter test kit. These test runs were composed of a number of HTTP requests that simulate the use of the system by a human user. Each test suite began with initiating a new Search and Browse session, and then ran through the actions that are necessary to complete the use case as described previously during the functionality test (section 2.7).

For these test suites it was determined how fast the system responded to each user request that is part of the use case, and how long it took to complete the total use case. Each test suite was run at least 100 times, to get average results for the response times. After that for each use case the system was subjected to a critical evaluation of its performance under high load, to determine

its scalability. Therefore the test suits were started concurrently with an increasing number of simultaneous runs. Again the response times were determined. It was also checked how reliable the system was under the high load, by verifying the results and noting errors.

3.3.5.2 Test log

```
Test 1:

Use Case: Browse Folder Collection

Description of Test:

- Call S&B GUI from user folder
- select collections from folder
- add collections to query

Results:
```

concurrent calls	1	5	10	15	20
loops	200	40	20	15	10
total calls	200	200	200	225	200
TASK:					
- Call S&B GUI	571ms	1151ms	1674ms	2108ms	2392ms
- Select Collection	543ms	1041ms	1630ms	2634ms	4814ms
- Add Collection	761ms	1508ms	2109ms	2462ms	2910ms
Total Use Case:	~2s	~4s	~5s	~7s	~10s
all responses o.k.	x	x	x	x	x

```
Test 2:

Use Case: Browse Personal Collection

Description of Test:

- Call S&B GUI from user folder
- select collections from personal set
- add collections to query

Results:
```

concurrent calls	1	5	10	15	20
loops	200	40	20	15	10
total calls	200	200	200	225	200
TASK:					
- Call S&B GUI	544ms	1068ms	1557ms	2294ms	2974ms
- Select Collection	501ms	1102ms	1749ms	1906ms	2130ms
- Add Collection	677ms	1293ms	2004ms	2730ms	3647ms
Total Use Case:	~2s	~3s	~5s	~7s	~9s
all responses o.k.	x	x	x	x	x

Test 3:

Use Case: Browse Folder Collection

Description of Test:

- Call S&B GUI from user folder
- select collections from system
- add collections to query

Results:

concurrent calls	1	5	10	15	20
loops	200	40	20	15	10
total calls	200	200	200	225	200
TASK:					
- Call S&B GUI	669ms	1015ms	1474ms	2263ms	3146ms
- Select Collection	456ms	689ms	997ms	1542ms	1891ms
- Add Collection	789ms	1250ms	1915ms	2532ms	3541ms
Total Use Case:	~2s	~3s	~4s	~6s	~9s
all responses o.k.	x	x	x	x	x

Test 4:

Use Case: Save Query

Description of Test:

- Call S&B GUI from user folder
- start new query
- add new condition (three times)
- name query
- save query

Results:

concurrent calls	1	5	10	15	20
loops	200	40	20	15	10
total calls	200	200	200	225	200
TASK:					
- Call S&B GUI	764ms	1158ms	1901ms	2705ms	2982ms
- Start Query	663ms	899ms	1498ms	2096ms	2156ms
- Add Condition	808ms	1166ms	1961ms	2751ms	3068ms
- Name Query	427ms	688ms	1269ms	1568ms	1956ms
- Save Query	995ms	2381ms	3106ms	3374ms	3991ms
Total Use Case:	~5s	~9s	~14s	~18s	~20s
all responses o.k.	x	x	x	x	x

Test 5:

Use Case: Get New For Folder / Save Results

Description of Test:

- Call S&B GUI from folder
- get new records
- save a result record to folder

Results:

concurrent calls	1	5	10	15	20
loops	200	40	20	15	10
total calls	200	200	200	225	200
TASK:					
- Call S&B GUI	602ms	648ms	1012ms	1793ms	2019ms
- Get New Records	3335ms	4903ms	6724ms	9789ms	10274ms
- Save Record	1073ms	1080ms	1198ms	1450ms	1826ms
Total Use Case:	~5s	~7s	~9s	~13s	~14s
all responses o.k.	-	-	-	-	-

Problems with Get New Records:

- NullPointerException
- Cyclades code #12904, java.io.IOException: Connection refused
- sometimes very long response time: >30sec

Test 6:

Use Case: Query Without Personalization

Description of Test:

- Call S&B GUI from user folder
- start new query
- add a new condition
- send query
- view a result record

Results:

concurrent calls	1	5	10	15	20
loops	100	20	10	7	6
total calls	100	100	100	105	120
TASK:					
- Call S&B GUI	553ms	891ms	1100ms	1848ms	2294ms

```
- Start New Query    |  518ms |   812ms |  1004ms |  1623ms |  1536ms
- Add Condition      |  609ms |   922ms |  1170ms |  1807ms |  1848ms
- Send Query         | 4827ms |  9297ms | 17284ms | 24212ms | 32941ms
- View Record        |  645ms |  1352ms |  2125ms |  3377ms |  4317ms
-------------------------------------------------------------------
Total Use Case:      |   ~4s  |   ~13s  |   ~23s  |   ~33s  |   ~43s
all responses o.k.   |    x   |     x   |   96 %  |   94 %  |  99 %

  - a small number of OutOfMemoryExceptions occured during the
    fourth task when the server was subjected to high load
  - searching is currently being reimplemented to take care of the
    discovered problems


Test 7:

Use Case: Query With Personalization

Description of Test:

- Call S&B GUI from user folder
- start new query
- add a new condition
- send a filtered query
- view a result record

Results:

concurrent calls     |     1   |     5   |    10   |    15   |    20
loops                |   100   |    20   |    10   |     7   |     5
total calls          |   100   |   100   |   100   |   105   |   100
-------------------------------------------------------------------
TASK:                |         |         |         |         |
- Call S&B GUI       |   532ms |   574ms |   763ms |  1217ms |  1401ms
- Start New Query    |   489ms |   499ms |   602ms |   990ms |  1003ms
- Add Condition      |   614ms |   657ms |   783ms |  1152ms |  1165ms
- Send Filt. Query   | 15570ms | 45285ms | 92045ms |135485ms |150789ms
- View Record        |   459ms |   503ms |   783ms |   983ms |   992ms
-------------------------------------------------------------------
Total Use Case:      |   ~17s  |   ~47s  |   ~95s  |  ~140s  |   ~155s
all responses o.k.   |     x   |     x   |     x   |     x   |     x
```

3.3.5.3 Test summary

Generally, the system showed efficiency in handling most of the tested use cases by responding to user input and requests within short time. With two exceptions no single response took more than a second under controlled conditions, and even under simulated heavy load the system responded on average within 4 seconds.

Both the sending of a query and the retrieving of new records for a given folder took longer than a second to complete even under controlled condition, which was to be expected. Both also showed a lack of stability in their performance, but to different degrees. Where with querying only a few errors crept in under high load, during the use case test for “Get New For Folder” a high number of errors occured that made reliable testing difficult.

Without a working Access Service and its search functionalities, it wasn't possible to use any of the search and browse related functions of the service. However, appropriate error messages were generated and the system didn't crash. Since the GUI of the Search and Browse Service can be called independently from the rest of the system, it was still be functional without a running Mediator Service.

Without a Collection Service, it wasn't possible to restrict searches to a smaller number of collections. Instead all searches automatically used all collections, resulting in longer search times. Searching could still be completed. A missing Filtering and Recommendation Service prevented the use of those functionalities related to personalized searching and browsing. As an enhancement to the current implementation it might be possible to default to unpersonalized searching in case of a missing FRS, instead of showing an error message.

Lastly, without a working Collaborative Work Service or when missing information about the current user and folder, all search and browse functionalities were still working stable, while menu options related to personalized browsing, or persistency of queries and results (saving and retrieving saved queries, getting new record for the folder, searching with personalization, and saving records) were hidden from the user.

3.4 Summary

During the performance testing phase of validating the CYCLADES system, a number of tests were done for all parts of the system, as well as the integrated system as a whole. These tests were based on the procedures proposed in deliverable 2.2.1, and included the efficiency evaluation of the system's methods, as well as stability and scalability tests.

All in all, the system behaved stable even under simulated stress for most use cases, performed efficiently, and scaled well. In the course of testing several minor problems, especially related to concurrent method calls, became evident, and were for the most parts fixed. New tests were done to verify the performance of the newly enhanced system, and these were documented in this report.

Appendix A

Terminology

- *archive*
  An *archive* is a set of records describing documents by metadata. An archive is uniquely identified by a string, called the *archive identifier*.
- *document*
  An arbitrary external file that can be stored in a project folder.
- *archive document*
  A resource that the user is interested in discovering in an archive. A document may be text (papers, reports, journals) or any other kind of media. Each document is described by one or more metadata records that are contained in and retrieved from open archives. Note that an archive need not contain documents. Depending on the organisation hosting the archive, it might as well contain only metadata records.
- *(metadata) record*
  The entities contained in an archive are *records*. Each such record consists of a set of attributes and values describing a document, according to a specific metadata schema. One document can be described by several metadata records using different schemas.
- *metadata schema*
  A set of attribute definitions, including attribute names, the type of each attribute.
- *collection*
  A set of records defined by a set of archives and an optional *filtering query* (i.e. a selection criterion) which uses only Dublin Core attributes. All the records from the given archives which satisfy the query are members of the collection defined. A collection may provide different metadata schemas and different search services for each schema.
  From the user's point of view, a collection is a set of documents with specific formats of search and browse operations associated with them. In this perspective, a collection is described by a set of criteria that specify which are the documents that belong to the collection (*membership condition*) and by the format of the search and browse operations.
- *community*
  A *community* is a set of users sharing a common (scientific, professional) background or view of the world. Within Cyclades, communities are characterized by a shared interest in documents and records. This common interest manifests itself as a hierarchy of common folders where records of common interest are stored and commented.
- *project*
  This is a group of scholars working together closely, presumably in a common project or field of interest. They do not only want to share records of interest, but possibly also other kinds of documents and modify them in a common workspace (i.e. in a common folder system).

- *folder*
  This is a container for metadata records and, in the case of project folders, possibly also documents. Collections can be associated to folders. This means that the user can choose by default from these collections when she formulates a query. Queries can be stored in folders, allowing the user to resubmit or edit them at any later time. Thus, a folder can also be seen as an environment corresponding to a certain topic, where the user is interested only in data concerning this topic.

- *private folder*
  A folder owned by a single user. This kind of folders can only be accessed by their owner. For others, they are invisible.

- *community folder*
  A folder owned by a community. This is a folder that can be accessed and possibly manipulated by several users that thus form a community (see above).

- *project folder*
  A folder owned by a project, i.e. accessible to the members of a project. Only project folders can contain documents, private and community folders can contain only records and queries.

- *subfolder*
  A *subfolder* is a folder that is contained in another folder called the *parent folder*. Each folder can maximally have one parent folder. If a subfolder is copied to another folder, then the folder and its whole contents are copied (not only a reference), thus creating a distinct new folder.

- *home folder*
  Each user has a special folder called the *home folder* which constitutes the root of the folder hierarchy that is visible to the user.

- *user*
  A person registered in the Cyclades environment. A user has an identity, an e-mail address, a password and a home folder. Users can have further attributes like affiliation, postal address, telephone number.

- *rating*
  A relevance value assigned to a record by a user. The user may rate a record directly by assigning it a certain value, or indirectly by choosing the record from a query result list and storing it in a folder. In the latter case, the system assumes the record to be relevant and assignes it a positive rating value automatically.