

Object Oriented Community- Information Tools

Cornelia Stinchcomb

neli@sni.net

1.0 Introduction

COMET, PAGE and Unidata are investigating development of a common strategy to support information sharing within their communities. This investigation has provoked discussion on the high level architecture of information technology support for community information sharing. An initial set of object oriented community-information tools will provide integrated support for community access lists, and possibly for access to community email archives. Development will serve as a test bed for design and deployment of distributed OO technologies whose use is anticipated in larger projects.

1.1 Document Content

This document summarizes requirements for the Object Oriented Community-Information Tools (OOCIT) project. It also contains an outline of the intended architecture (Section 9.0 on page 20) and a few notes on user interface design (Section 10.0 on page 21).

1.2 References

Project Documentation

- Object Oriented Community Information Tools [project plan](#).

Web Resources

- UCAR's [Community Access](#) web pages.
- LDAP is key to sharing address information with email packages.
 - [University of Michigan LDAP](#).
 - [LDAP Roadmap and FAQ](#) (Stanford)
 - [SunWorld's LDAP overview](#)
 - [LDAP World](#) page maintained by a commercial developer.
 - [DevEdge Online](#): Another overview.
 - [IDM overview](#).
- [JNDI](#) (Java Naming and Directory Services) documentation.
- Sun Microsystems' [JavaMail](#) documentation and API.

- [Java Servlets](#) may be used to communicate between the client and application server instead of CGI. The [API specification](#) is here.
- The [Advanced-Java mailing list archives](#) may be found on the xcf.berkeley.edu/lists.html page. This page is also interesting because it provides access to archives and subscription services for several mailing lists.
- A [Majordomo FAQ](#), useful for those with less experience with this package.

Books

- Nancy Wilkinson: *Using CRC Cards: An Informal Approach to Object Oriented Development*. SIGS Books, 1995.
- Martin Fowler, with Kendall Scott: *UML Distilled*. Addison-Wesley, 1997.
- Ivar Jacobson et. al.: *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1994.
- Martin Fowler: *Analysis Patterns: Reusable Object Models*. Addison-Wesley, 1997.
- Erich Gamma et. al.: *Design Patterns: Elements of Reusable Object Oriented Software*. Addison-Wesley, 1994.
- Timothy A. Howes and Mark C. Smith, *LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol*. Macmillan Technical Publishing, 1997.
- Ben Shneiderman, *Designing the User Interface, 3rd Ed.* Addison-Wesley, 1998.
- Hugh Beyer and Karen Holtzblatt, *Contextual Design*. Morgan Kaufmann, 1998.

Magazines and journals

- *JOOP: the Journal of Object Oriented Programming*. SIGS Publications. <http://www.sigs.com>
- *Java Report*. SIGS Publications. <http://www.javareport.com>
- *Object Magazine*. SIGS Publications. <http://www.sigs.com>

1.3 Acronyms

The following acronyms are used in this document

TABLE 1. Acronyms

Acronym	Description
LDAP	Lightweight Directory Access Protocol
ODBMS	Object Database Management System. An object repository.
OOCIT	Object Oriented Community-Information Tools
JNDI	Java Naming and Directory Interface

1.4 About this document

The remainder of this document is organized as follows:

- *Product overview*: system objectives, assumptions and dependencies.
- *Domain object model*: a conceptual object-oriented analysis of the domain, focused on domain entities.
- *Use cases*: provide an analysis of the system functional requirements using a task oriented approach.
- *Non-functional requirements*: performance, scalability, extensibility, availability, reliability, portability, usability, security, documentation, testability, localization.
- *System constraints*: cost, schedule, deployment platforms, tools.
- *Priorities*: feature development priorities will be catalogued during analysis.

2.0 Product Overview

2.1 System Objectives

Provide integrated support for community access lists, and possibly for access to community email archives.

Support both voluntary and role-based (organizational) communities.

2.2 Assumptions

System should be UCAR-centric and accessible to the community. Some functions will be reserved for administrators. UCAR staff may have special privileges.

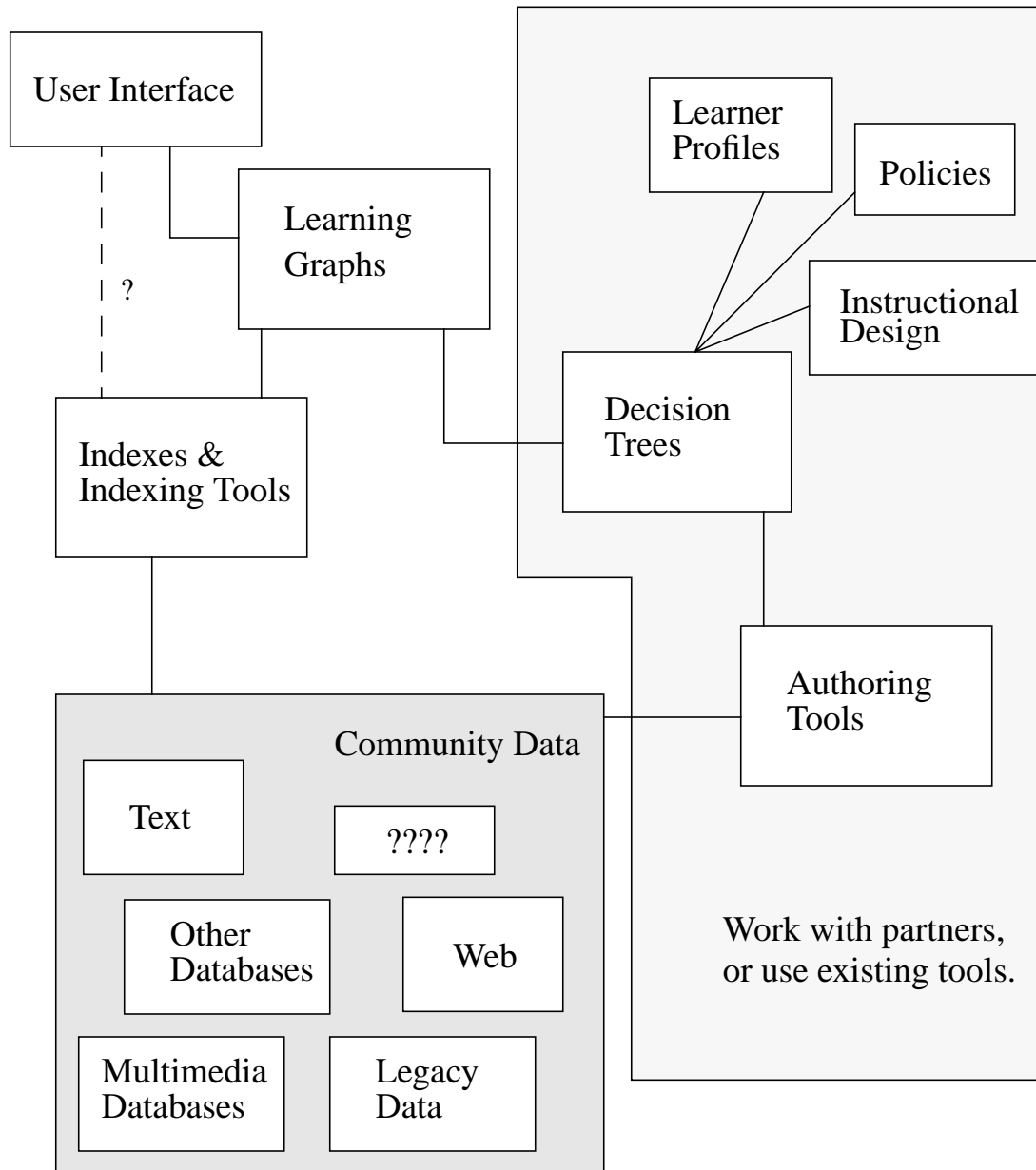
The new system should be at least as functional as the current system at the time of deployment.

2.3 Dependencies

3.0 Domain Object Model

3.1 Context

FIGURE 1. Community Information Sharing: Context

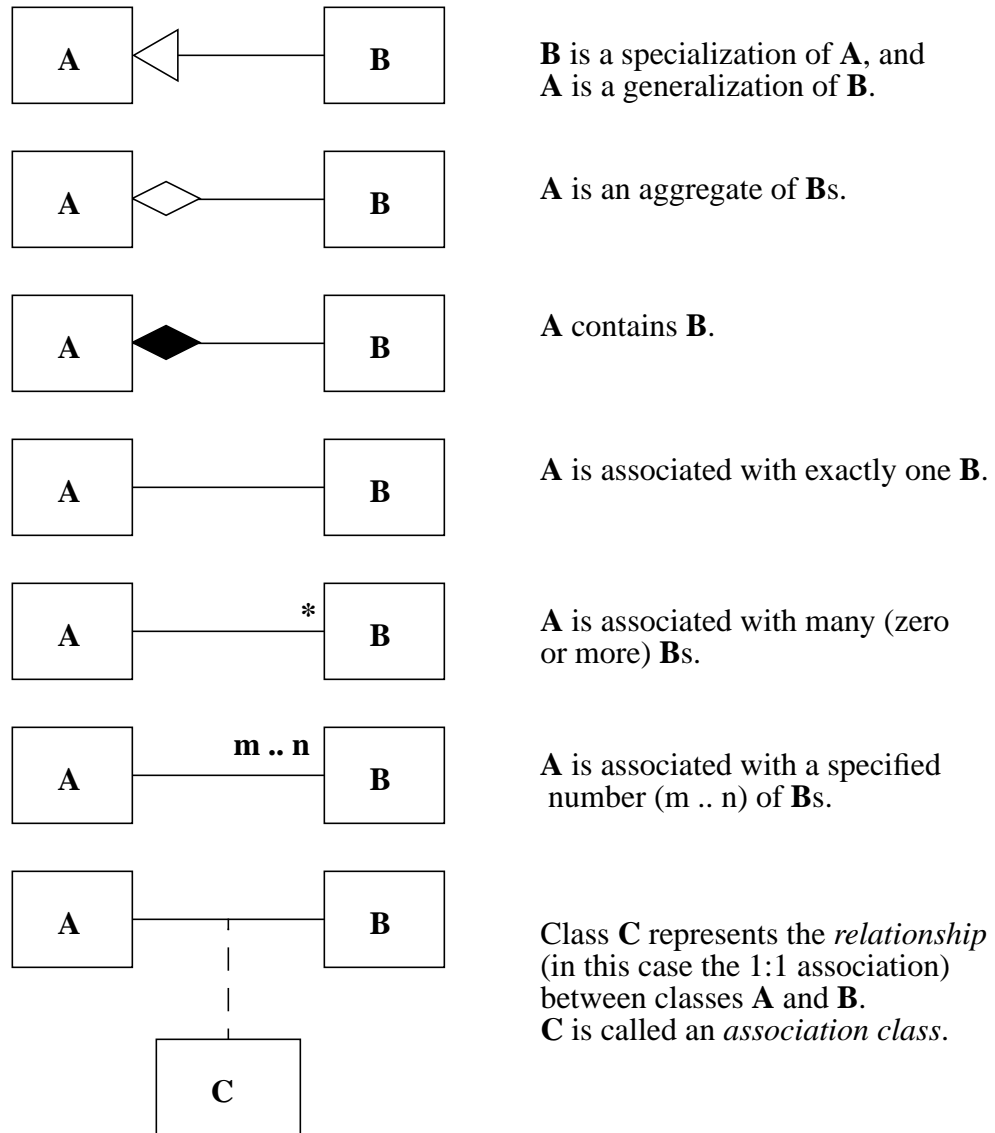


It was suggested that significant innovation could occur in the areas of indexes and indexing tools, and support for learning graphs.

3.2 Overview

We are using a subset of the UML (Unified Modeling Language; www.rational.com) to describe the OOCIT domain.

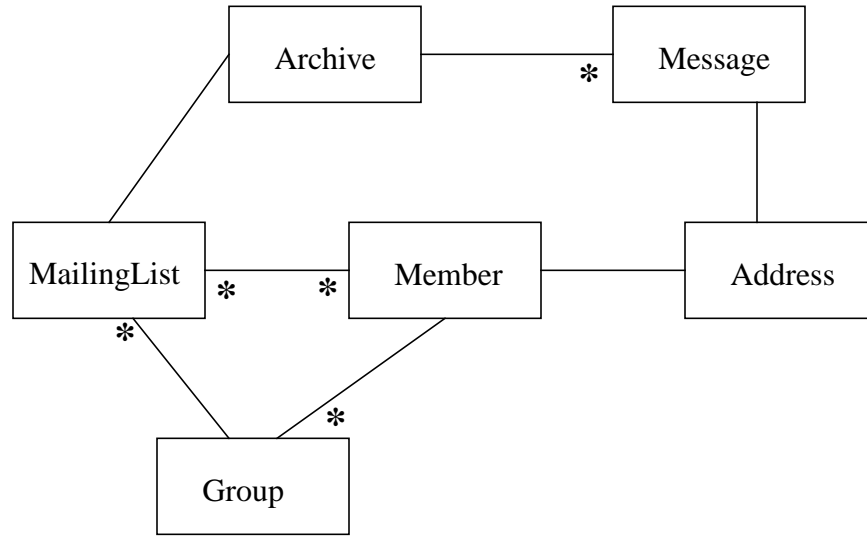
FIGURE 2. UML Notation Summary



3.3 Classes

This section should be updated with attributes and behaviors discovered during user interface design.

FIGURE 3. Overview of Associations



3.3.1 Member

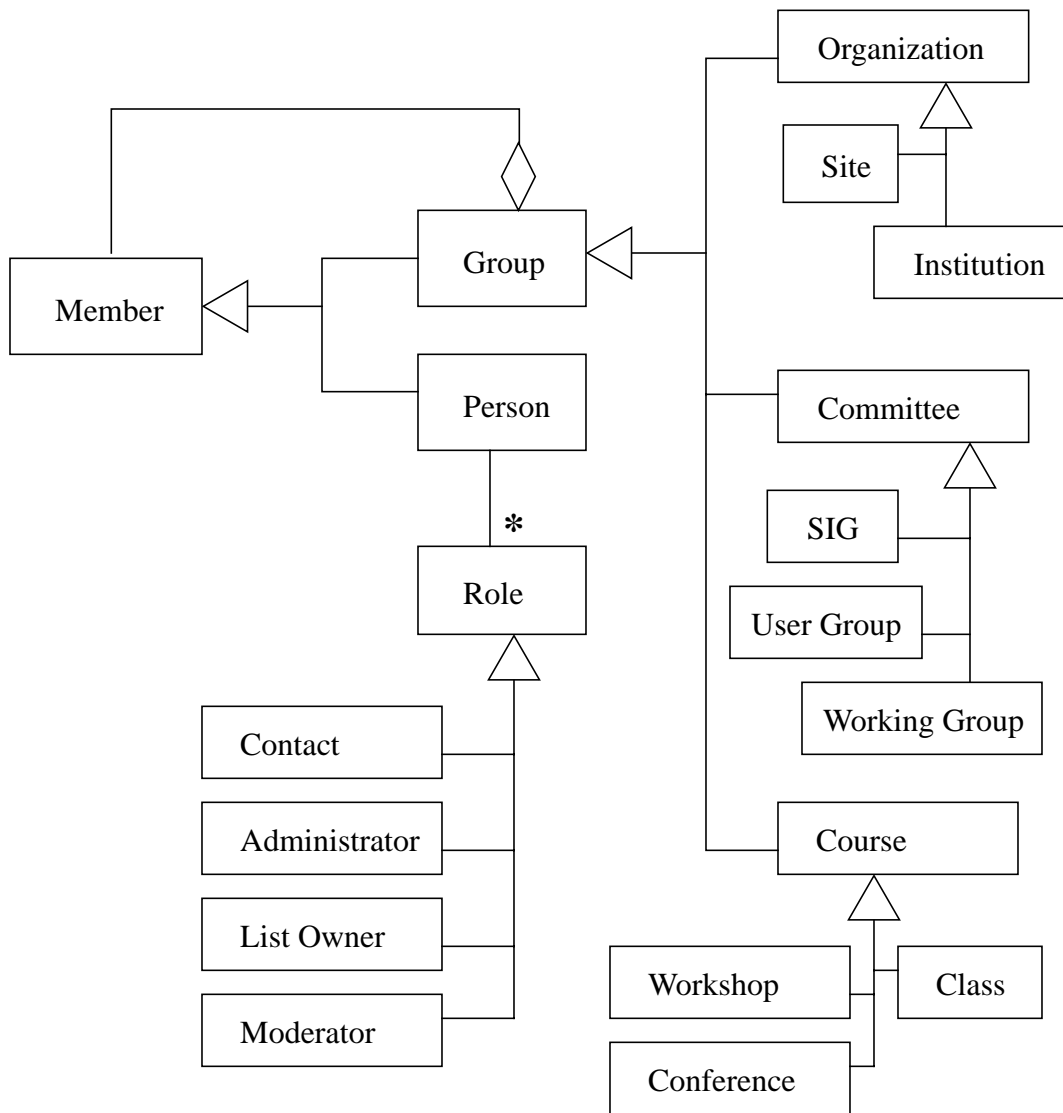


TABLE 2. Site Data

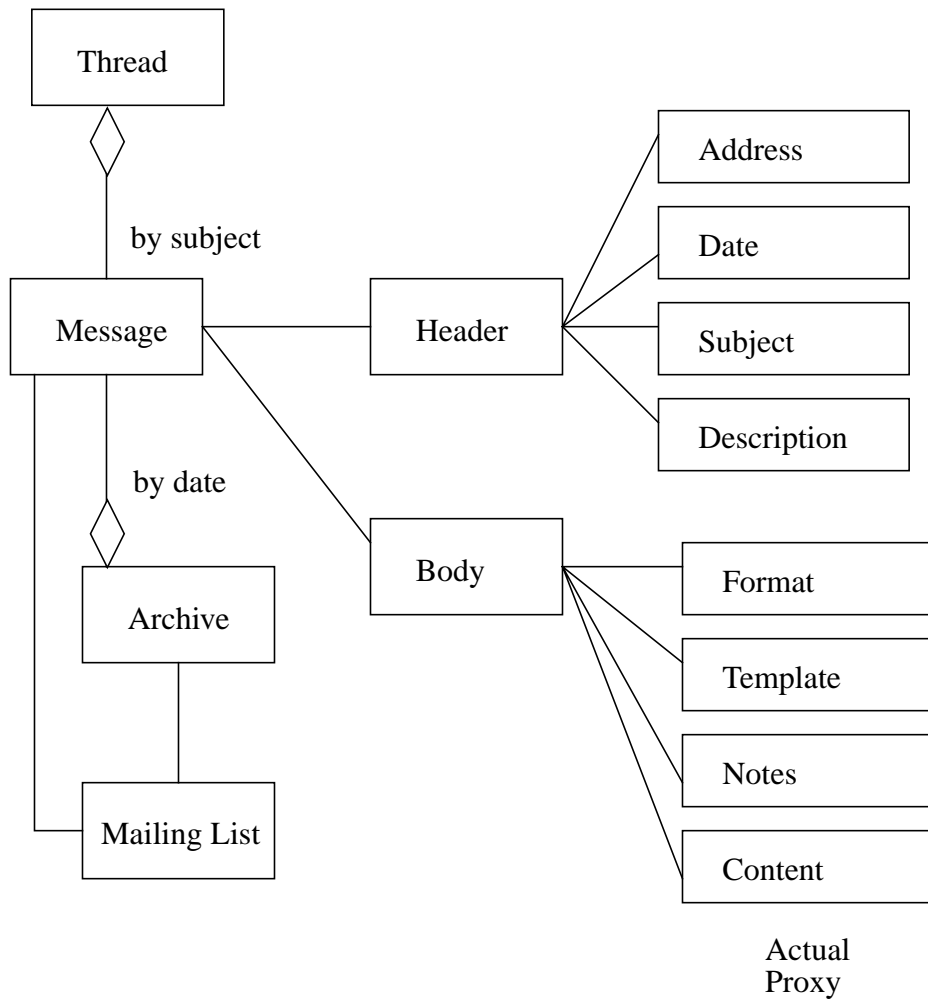
Data	Description
Real-time data profile	Profile contains a list of data sets plus statistics for each set.
Software licenses	List of Licenses
Computing platforms	List of PlatformDescriptions
Technical contact	Role
License contact	Role
Administrative contact	Role
Contract contact	Role

TABLE 2. Site Data

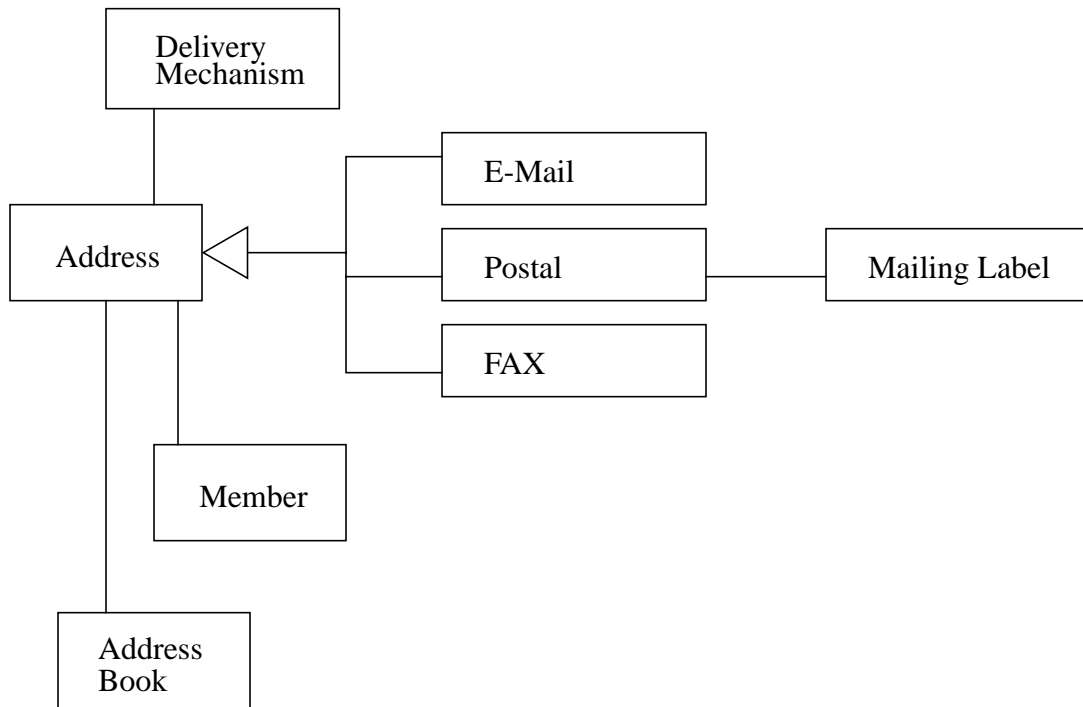
Data	Description
Site web servers: URLs for more general site information	Link to group's primary URL
List of references	List of associated URLs

Site data has been extracted from the Use Case descriptions. It will be used as a basis for Group and Site attributes.

3.3.2 Message



3.3.3 Address



3.3.4 Mailing List

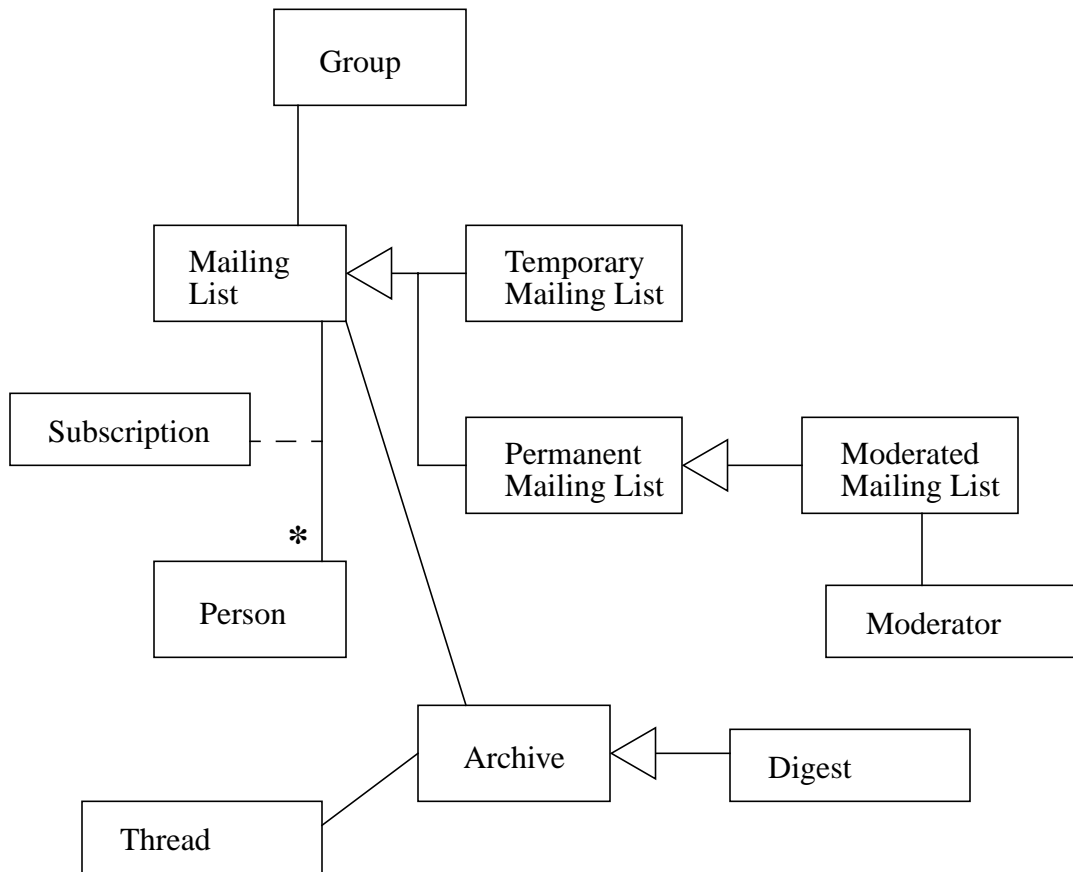
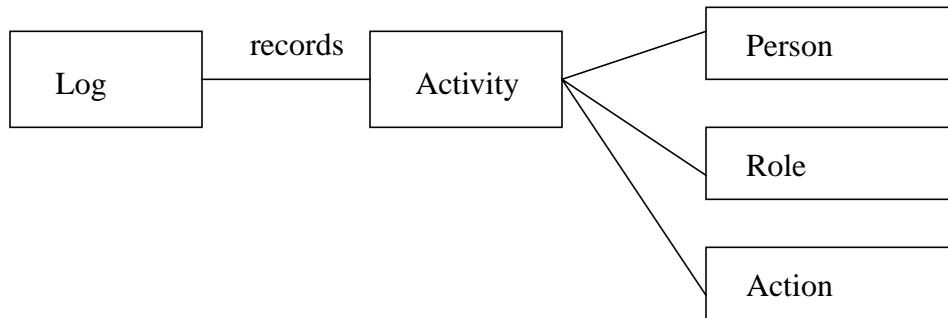


TABLE 3. MailingList Data

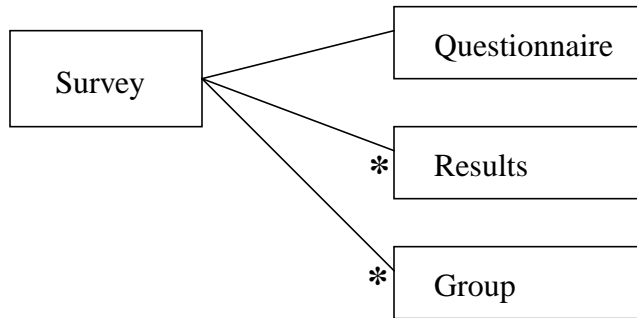
Data Item	Description
description	Text description of list purpose/charter
owner	Person
moderator-list	List of Persons filling Moderator role
member-list	List of subscribed Members
password	
security policy	

Initial MailingList data was extracted from the Use Case descriptions. This list is not complete.

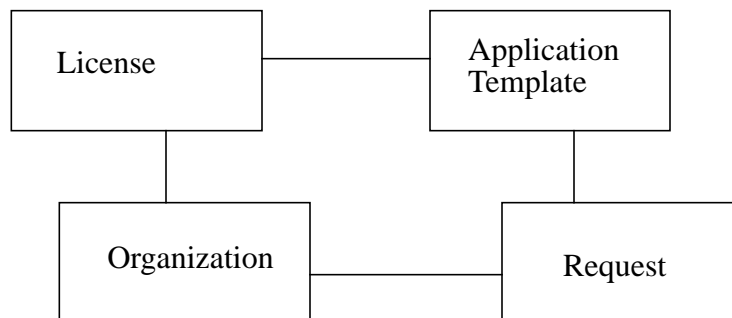
3.3.5 Log



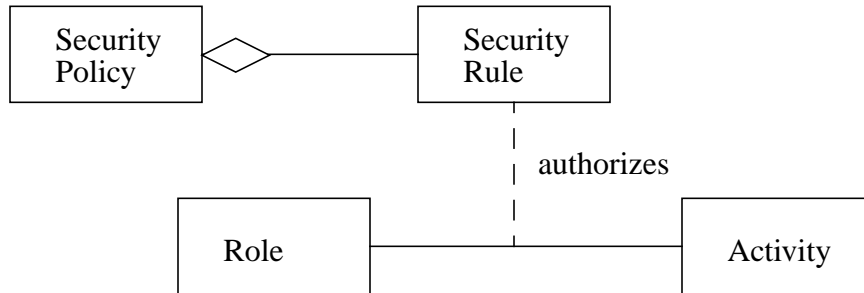
3.3.6 Survey



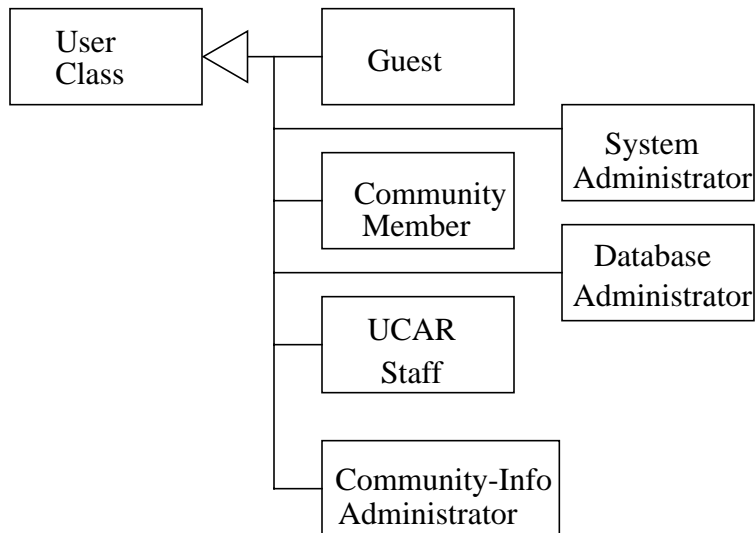
3.3.7 License



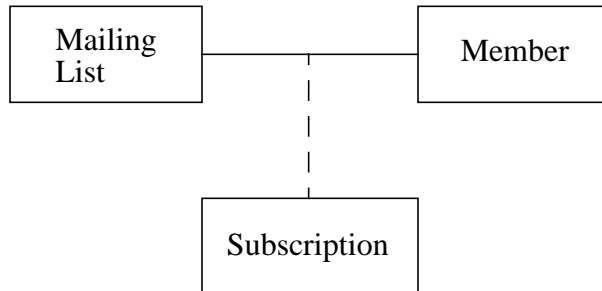
3.3.8 Security Policy



3.3.9 User Class



3.3.10 Subscription



Subscription Behaviors:

- Create new subscription
- Remove subscription
- Suspend
- Resume
- Receive digest
- Receive messages

4.0 Use Cases

The use cases in this section were brainstormed on 9 March 98. Initial priority numbers were added on 2 April 98, and are subject to change.

Bold items are used in the domain object model. Initial priorities precede the use case description, in [] square brackets.

1. [2] Access a community **address book** through (some) email packages.
2. [1] Send **email** to community **members**
3. [2] Send **postal mail** to community members: generate **postal mail labels**
4. [3] Send a **fax** to community members
5. Add/delete/modify community **member information**
 - [2] by member
 - [1] by administrator
6. [2] Add/modify a {list, site, member} attribute, type of data, or class
7. Create a new **community** sub-list based on criteria

- [1] find members
- [1] create a new **persistent list**
- [1] create a new **temporary list**
- [1] merge lists and remove duplicates
- [1] profile a community
- [1] contact list members
- 8. Support a **moderated list**
 - [3] Mail approved by one or more moderators
 - [2] Only list members may post
- 9. [3] Approve a **message** to be sent to a **list**
- 10. [2] Archive messages
- 11. Access **message archives**
 - [2] Simplest archive access
 - [3] Improved access with good search technology
- 12. [1] Community member subscribes to one or more lists
- 13. [1] Add/modify/delete organization info: “**site data**” Such data includes:
 - disciplines served
 - real-time data profile**, including **type of data** and, for each type, data **statistics**.
 - software licenses: licensed packages**, and dates
 - computing **platforms**
 - technical **contact**
 - license contact
 - administrative contact
 - contract contact
 - site web servers: **URLs** for more general site information
 - list of references
- 14. Display **member information**
 - [1] for a selected member
 - [2] generate a custom **report** on a set of members
- 15. [1] Search for members based on member information. Member information includes:
 - expertise
 - interests
 - location
 - title

- role
- subscribed mailing lists
- contact information: phone, email, fax, postal addresses
- current committees and other **roles**
- preferred **message delivery method**
- URLs for more member information
- 16. [1-3] Support an **authorization** mechanism. Simplest mechanism first, then improve functionality.
 - Authorize **user action**
 - Approve subscription **request**
- 17. [1-3] Set/modify **security policy** for a member, an organization, a list, a community. Simplest mechanism first, then improve functionality.
 - Add/remove **security rule**
- 18. Conduct **survey**
 - [2] Issue survey
 - [2] Collect **results**
 - [3] Report results
 - [3] Update profile data from survey results
- 19. [1-3] Log **activity**. Simplest access first, improve later
 - subscriptions
 - member access
 - administrator activity
- 20. [3] Support **digests**
 - set/modify digest period
- 21. [1] Add/modify **list data**, including
 - description
 - owner
 - moderator-list
 - member-list
 - password
 - security policy
- 22. [2] Populate database
 - from existing data
 - by sifting user interactions
- 23. [3] Support the licensing process
 - provide site **license application** (from the web, or order hard copy)

update member/site information from completed license application
generate new **license** from completed license application

24. [3] Generate message from **message template**

5.0 Non-Functional Requirements

5.1 Performance

Reasonable interactive performance is expected for queries.

5.2 Scalability

It is important that the community information framework scale to accommodate heterogeneous data sources and large data objects.

5.3 Availability

The OOCIT system should be available most of the time, but scheduled downtime for system upgrades and other administrative activities is acceptable. Such activities should occur, if possible, during off-hours for the US. This indicates an availability of 7x22 or so.

MTTR (mean time to repair) after a failure is suggested to be 4 hours.

5.4 Reliability

Core functionality must be reliable. New features that may be less reliable should be labeled to the user as such.

5.5 Operating Environment

The OOCIT toolset is expected to operate in a WWW environment.

5.5.1 Deployment Platform

Users will use a web-based client: internet browser with applets.

A SUN Solaris server will host the application (domain-level) tier and will also host the object repository.

The Java programming language will be used for implementation.

5.5.2 Portability

In general, a portable design is a better design. A heterogenous set of client browsers is expected for Guests and Members. A less heterogenous set of browsers may be used by Administrators.

On the server side, connections to external services and data sources should be encapsulated. Operating-system dependent code should be minimized and encapsulated if necessary.

5.6 Usability

5.6.1 User Classes

The following classes of users have been identified. Specific use cases may be targeted towards particular user classes. Users in a particular class will have permission to undertake specific actions.

Guest

A guest has not identified themselves (logged in) to the OOCIT system.

Community member

A community member has registered themselves with the OOCIT system and has logged it for the current session. A member may modify only their own data.

UCAR staff

UCAR staff may have special privileges not permitted for the wider user community. Specific privileges have not been identified.

Community-Information Administrator

Permitted to configure the OOCIT tools, and to modify member and other data that may not be modified by members of other user classes.

System Administrator

Root-level permissions on the server.

Database Administrator

Special permissions for modifying the database itself.

5.6.2 Usability testing

Testing of the interface by each class of users is desired prior to deployment.

5.7 Extensibility

The OOCIT toolset is intended to serve as a foundation for significant further development. Integration of additional data sources/external interfaces is expected and should be planned for.

Extensions to the object model can also be expected.

5.8 Security

Concerns:

- Personal information
- Spam

5.9 Documentation

5.9.1 User Documentation

An on-line help system is required. It should include

- Field level help
- Screen level help
- A searchable user's guide

5.9.2 Internal documentation

Since new developers will likely be added to the original team over time, good internal documentation would be useful. However, we have spent a lot of front end time generating primarily documentation; it's time for code.

5.10 System Test

A set of client browsers plus one server architecture should be sufficient. Alpha test should include users of each class and may be restricted to members of the design team. Beta test deployment throughout UCAR is recommended prior to general deployment.

A brief test plan for each user class should be written before the start of system test.

5.11 Localization

English-only for now. Better localization support (I think it will be a catalog-based model) will be available in Java 1.2.

6.0 System Constraints

6.1 Development constraints

6.2 Deployment constraints

7.0 Priorities

Use cases are annotated with current priority numbers.

8.0 Issues

The following concerns have been raised during analysis:

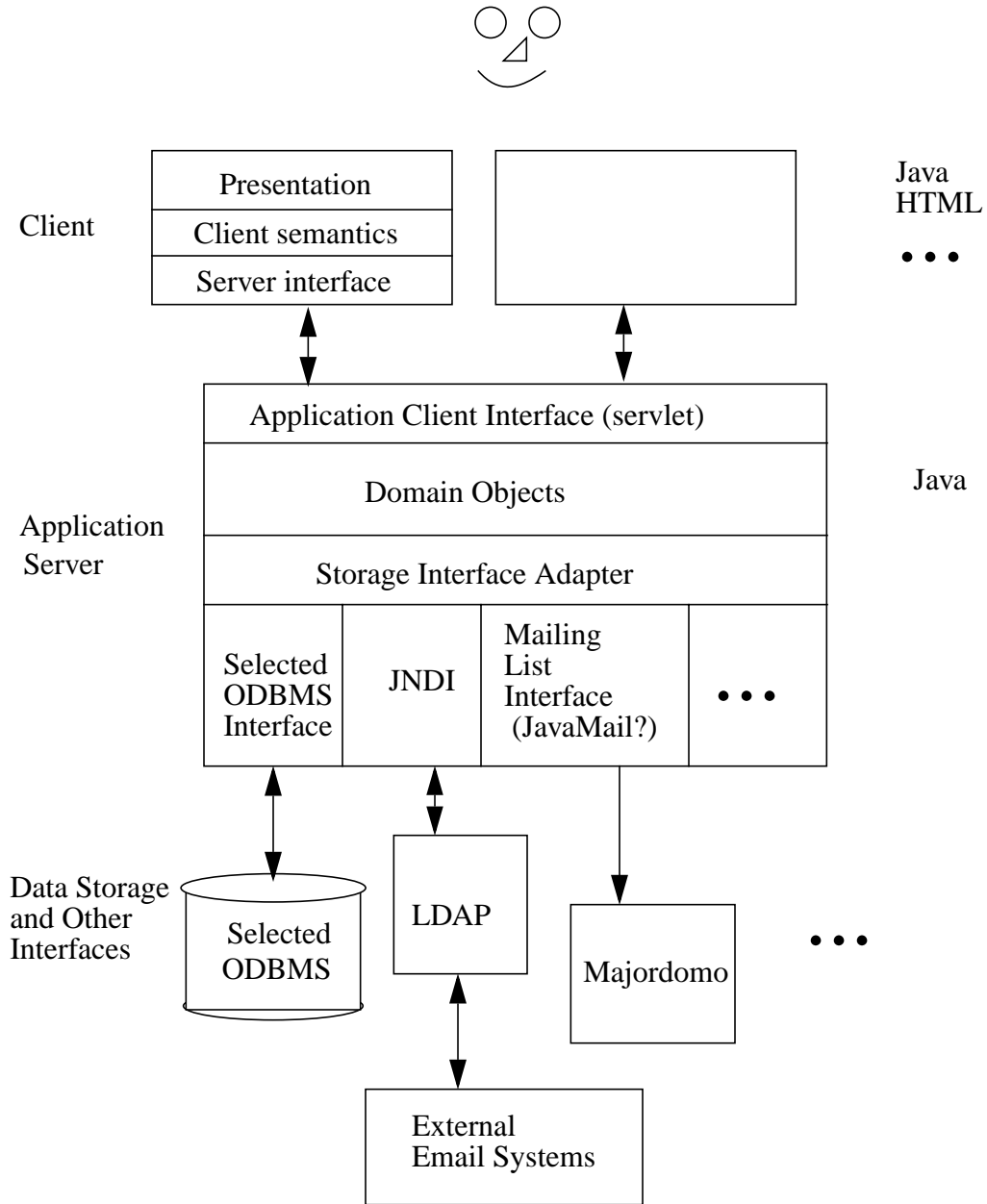
- *Out-of-date data*: Data should be kept in its most current state to the extent possible. If data can be linked to rather than maintained internally, we may have a better chance at keeping it up to date. Automatic methods of keeping data updated (where the intervention of a human administrator is not required) should be investigated and implemented where possible.
- *PC Interoperability*: “Front-office” administrative staff is Windows-based and want their software to interoperate well with the tools they are used to using, such as Microsoft Office.

These issues are related to the domain model:

- *Roles and User Classes*: Several possible relationships between roles and user classes exist. Which is most useful?

9.0 System Architecture

9.1 Overview



9.2 External Interfaces

9.2.1 Object Repository

An evaluation questionnaire has been prepared, and a matrix of vendor answers is being constructed. Initial evaluations of Jasmine and Gemstone are underway. Versant and Objectivity may also be evaluated.

The architecture accommodates multiple simultaneous repositories, although only a single repository is shown in the overview.

9.2.2 JNDI and LDAP

9.2.3 Majordomo

We don't intend to reinvent the wheel. This interface may use ASCII files.

9.2.4 JavaMail

10.0 User Interface

The user interface design work undertaken to date has been done using lo-fidelity paper prototypes. These drawings were copied and distributed to team members but have not been included in this document.

10.1 The design process

An iterative process was used to design a core set of screens:

1. Review object model
2. Revise object model
3. Select a use case, and use it to focus on a set of screens.
4. Create or revise overview, search and detail screens.
5. Draw or update the navigation diagram that illustrates connections between screens.
6. Review other input and annotate current design.
7. Begin another iteration.

The user interface design process highlighted issues in the domain object model and forced us to better enumerate object attributes and behaviors. Navigation diagrams highlight object collaborations.