

# **System and Software Architecture Description (SSAD)**

**Digital Archive Usage Analysis**

**Team #7**

**Bo Lee  
Genesan Kim  
Maks Krivokon  
Vu Nguyen**

**04/25/2005**



# Version History

Date	Author	Version	Changes made
9/28/04	Hsiao-Han Huang Pei-Han Li	1.0	<ul style="list-style-type: none"> <li>• Early Section</li> </ul>
10/08/04	Hsiao-Han Huang Pei-Han Li	1.1	<ul style="list-style-type: none"> <li>• Correction and add diagrams and tables</li> </ul>
10/14/04	Hsiao-Han Huang Pei-Han Li	1.2	<ul style="list-style-type: none"> <li>• Finish the section 3</li> </ul>
10/24/04	Hsiao-Han Huang Pei-Han Li	2.0	<ul style="list-style-type: none"> <li>• Correction and add 3.5</li> </ul>
11/17/04	Hsiao-Han Huang Pei-Han Li	3.1	<ul style="list-style-type: none"> <li>• Correct section 2 section 3 and add section4</li> </ul>
12/6/04	Hsiao-Han Huang Pei-Han Li	4.0	<ul style="list-style-type: none"> <li>• Complete SSAD at LCA stage</li> </ul>
02/09/05	Maks Krivokon	5.0	<ul style="list-style-type: none"> <li>• Rewritten section 3</li> </ul>
02/25/05	Maks Krivokon	7.0	<ul style="list-style-type: none"> <li>• Rewritten section 4 – see 1.4 Change Summary</li> </ul>
04/04/05	Maks Krivokon	8.0	<ul style="list-style-type: none"> <li>• Incorporated feedback from graders and IV&amp;V reviewers – see 1.4 for change summary items 7 - Last</li> </ul>
04/25/05	Maks Krivokon	9.0	<ul style="list-style-type: none"> <li>• Rewritten the document to correspond to the system as built</li> </ul>

# Table of Contents

**SYSTEM AND SOFTWARE ARCHITECTURE DESCRIPTION (SSAD) ..... I**

**VERSION HISTORY ..... III**

**TABLE OF CONTENTS.....IV**

**TABLE OF TABLES .....VI**

**TABLE OF FIGURES .....VII**

1. Introduction..... 1

    1.1 Purpose of the SSAD Document..... 1

    1.2 Standards and Conventions..... 1

    1.3 References..... 1

    1.4 Change Summary..... 2

2. System Analysis..... 4

    2.1 Structure..... 4

    2.2 Artifacts & Information ..... 5

    2.3 Behavior..... 6

    2.4 L.O.S. Goals..... 11

    2.5 Rules ..... 12

3. Architecture Design & Analysis ..... 13

    3.1 Structure..... 13

    3.2 Analysis Classes..... 31

    3.3 Behavior..... 33

    3.4 L.O.S., Projected..... 38

    3.5 Architectural Styles, Patterns & Frameworks..... 39

4. Implementation Design..... 40

    4.1 Structure..... 40

    4.2 Behavior..... 64

    4.3 L.O.S..... 68

    4.4 Patterns & Frameworks..... 68

    4.5 Project Artifacts ..... 69

        <<source>>  
        alg.h ..... 69

        <<source>>      <<source>>  
        main.cc          H3Viewer.cc ..... 70

5. Glossary for System Analysis and Design..... 71

6. Appendices..... 72



# Table of Tables

<i>Table 1 Use-case description: Examine analysis results</i> .....	<b>Error! Bookmark not defined.</b>
<i>Table 2 Typical Course of Action: Analyze log data</i> .....	<b>Error! Bookmark not defined.</b>
<i>Table 3 Exceptional Course of Action: Analyze log data</i> .....	<b>Error! Bookmark not defined.</b>
<i>Table 4 Use-case description: Examine analysis results</i> .....	11
<i>Table 5 Typical course of action: Examine analysis results</i> .....	<b>Error! Bookmark not defined.</b>
<i>Table 6 Exceptional course of action: Examine analysis results</i> .....	<b>Error! Bookmark not defined.</b>
<i>Table 7 Available Capabilities &amp; Processes in Operational Mode</i> .....	11
<i>Table 8 L.O.S. goals</i> .....	12
<i>Table 9 Generate object tree description</i> .....	23
<i>Table 10 Typical course of action: Generate object tetree</i> .....	<b>Error! Bookmark not defined.</b>
<i>Table 11 Exceptional course of action: Generate object tree</i> .....	<b>Error! Bookmark not defined.</b>
<i>Table 12 Browse object tree description</i> .....	28
<i>Table 13 Generate relations implementation</i> .....	34
<i>Table 14 Browse object tree description</i> .....	38
<i>Table 21 L.O.S projected</i> .....	39

# Table of Figures

<i>Figure 1 System Static Structure diagram</i> .....	4
<i>Figure 2 System Collaboration diagram</i> .....	4
<i>Figure 3 Artifact Model</i> .....	5
<i>Figure 4 System Capability Realization</i> .....	6
<i>Figure 5 System processes and actors</i> .....	7
<i>Figure 6 Analyze log data Activity diagram</i> .....	8
<i>Figure 7 Examine analysis results activity diagram</i> .....	10
<i>Figure 8 Hardware classifier model</i> .....	14
<i>Figure 9 Software classifier model</i> .....	14
<i>Figure 10 Standard configuration hardware</i> .....	15
<i>Figure 11 Standard configuration software</i> .....	15
<i>Figure 12 Performance configuration hardware</i> .....	16
<i>Figure 13 Performance configuration Mainframe software</i> .....	16
<i>Figure 14 Performance configuration Workstation software</i> .....	16
<i>Figure 15 Relation generator interface</i> .....	17
<i>Figure 16 Relations generator processes</i> .....	18
<i>Figure 17 Generate relations Activity diagram</i> .....	19
<i>Figure 18 Generate relations use-case description</i> .....	20
<i>Figure 19 Typical course of action: Generate relations</i> .....	<b>Error! Bookmark not defined.</b>
<i>Figure 20 Exceptional course of action: Generate relations</i> .....	<b>Error! Bookmark not defined.</b>
<i>Figure 21 Tree generator interface</i> .....	21
<i>Figure 22 Tree generator processes</i> .....	22
<i>Figure 23 Generate object tree Activity diagram</i> .....	23
<i>Figure 24 Viewer operations</i> .....	25

<i>Figure 25 Viewer processes.....</i>	<i>26</i>
<i>Figure 26 Browse object tree Activity diagram.....</i>	<i>27</i>
<i>Figure 27 Analysis classes.....</i>	<i>32</i>
<i>Figure 28 System process implementation .....</i>	<i>33</i>
<i>Figure 29 Generate object relations implementation.....</i>	<i>34</i>
<i>Figure 30 Generate relations description .....</i>	<i>35</i>
<i>Figure 31 Generate object tree implementation.....</i>	<i>35</i>
<i>Figure 32 Browse object tree implementation.....</i>	<i>37</i>
<i>Figure 33 Hardware Classifier Model .....</i>	<i>40</i>
<i>Figure 34 Software classifier model.....</i>	<i>41</i>
<i>Figure 35 Hardware components used.....</i>	<i>41</i>
<i>Figure 36 Software components on Unix Mainframe.....</i>	<i>42</i>
<i>Figure 37 Software components on Mac OS X Workstation .....</i>	<i>42</i>
<i>Figure 38 log2mcl Internal Architecture.....</i>	<i>44</i>
<i>Figure 39 mcl2hv Internal Architecture .....</i>	<i>45</i>
<i>Figure 40 h3viewer Internal Architecture .....</i>	<i>46</i>
<i>Figure 41 Generate object relation realization.....</i>	<i>65</i>
<i>Figure 42 Generate object tree realization .....</i>	<i>66</i>
<i>Figure 43 Browse object tree realization .....</i>	<i>68</i>
<i>Figure 44 Project directory structure.....</i>	<i>69</i>
<i>Figure 45 log2mcl directory structure.....</i>	<i>69</i>
<i>Figure 46 log2mcl/src modified files.....</i>	<i>69</i>
<i>Figure 47 mcl2hv directory structure.....</i>	<i>69</i>
<i>Figure 48 mcl2hv/src modified contents.....</i>	<i>69</i>
<i>Figure 49 mcl2hv/include modified contents.....</i>	<i>69</i>

*Figure 50 h3viewer directory structure*.....70

*Figure 51 h3viewer/src modified contents*.....70

*Figure 52 h3viewer/include modified contents*.....70

# 1. Introduction

## 1.1 Purpose of the SSAD Document

This document is a refinement of architecture description compiled during the inception phase. The purpose of this document is to bring all the necessary details into architecture model and make it implementation specific. This document provides clear and sufficient description of system's structure and design and thus makes transition to implementation phase more efficient.

Intended audience of this document is as follows:

- development team should follow architecture specified in this document while constructing the system
- client can get in-depth insight into how the system works and what are functionality requirements
- maintainers can get better understanding of system structure that could facilitate debugging and extending the system.

## 1.2 Standards and Conventions

- Standard
  - MBASE Guideline for 577b version 2.4.2
- Notation
  - UML: version 1.4
- Naming Conventions
  - Components and object are Nouns.
  - Behaviors and Operations are Verbs.

## 1.3 References

MBASE Guidelines version 2.4.2

[http://sunset.usc.edu/classes/cs577a\\_2004/guidelines/MBASE\\_Guidelines\\_v2.4.1.pdf](http://sunset.usc.edu/classes/cs577a_2004/guidelines/MBASE_Guidelines_v2.4.1.pdf)

MBASE Electronic Process Guide

<http://cse.usc.edu/research/MBASE/EPG>

USC Information Services Division

<http://www.usc.edu/isd/about/about.html>

Fall 2004 CS 577a Project #7 Description

[http://sunset.usc.edu/classes/cs577a\\_2004/projects/description/project7.htm](http://sunset.usc.edu/classes/cs577a_2004/projects/description/project7.htm)

UML Guidelines

[http://sunset.usc.edu/classes/cs577a\\_2003/coursenotes/ep/Introduction\\_to\\_UML.pdf](http://sunset.usc.edu/classes/cs577a_2003/coursenotes/ep/Introduction_to_UML.pdf)

CS577a – Software Engineering I website

[http://sunset.usc.edu/classes/cs577a\\_2004/](http://sunset.usc.edu/classes/cs577a_2004/)

Fall 2003 CS 577a Project #8 LCO portion of OCD

[http://ebase.usc.edu/eservices/cs577a\\_2003/team08a/LCO/OCD\\_LCO\\_F03a\\_T08.pdf](http://ebase.usc.edu/eservices/cs577a_2003/team08a/LCO/OCD_LCO_F03a_T08.pdf)

Fall 2003 CS 577a Project #8 LCO portion of SSAD

[http://www-scf.usc.edu/%7Ecsci577/www/team8a/LCO/SSAD\\_LCO\\_F03a\\_T08.doc](http://www-scf.usc.edu/%7Ecsci577/www/team8a/LCO/SSAD_LCO_F03a_T08.doc)

Fall 2003 CS 577a Project #22 LCO portion of SSAD

[http://www-scf.usc.edu/%7Ecsci577/www/team22a/LCO/SSAD\\_LCO\\_F03a\\_T22.doc](http://www-scf.usc.edu/%7Ecsci577/www/team22a/LCO/SSAD_LCO_F03a_T22.doc)

Fall 2003 CS 577a Project #15 LCO portion of SSAD

[http://www-scf.usc.edu/%7Ecsci577/www/team15a/LCO/SSAD/SSAD\\_LCO\\_F03a\\_T15.doc](http://www-scf.usc.edu/%7Ecsci577/www/team15a/LCO/SSAD/SSAD_LCO_F03a_T15.doc)

Fall 2005 CS 577b Project #7 OCD Document

[http://seacliff.usc.edu/~team7b/RLCA/OCD\\_RLCA\\_S05b\\_T07\\_V7.0.doc](http://seacliff.usc.edu/~team7b/RLCA/OCD_RLCA_S05b_T07_V7.0.doc)

Fall 2005 CS 577b Project #7 SSRD Document

[http://seacliff.usc.edu/~team7b/RLCA/SSRD\\_RLCA\\_S05b\\_T07\\_V2.2.doc](http://seacliff.usc.edu/~team7b/RLCA/SSRD_RLCA_S05b_T07_V2.2.doc)

Fall 2005 CS 577b Project #7 UML Model

[http://seacliff.usc.edu/~team7b/RLCA/UML\\_Model\\_RLCA\\_S05b\\_T07\\_V7.0.mdl](http://seacliff.usc.edu/~team7b/RLCA/UML_Model_RLCA_S05b_T07_V7.0.mdl)

## 1.4 Change Summary

No	Change	Rationale	Sections Affected
1	removed description for business workers	system has one generic user	2.1
2	introduced new system artifacts: Item usage, Item tree, Item graph	consistency with system processes which operate on these artifacts	2.2, 2.3.1
3	Replaced 3 system processes with 6 new processes that map to System Capabilities defined in OCD and cover behavior of all System components	Consistency with OCD Cover behaviors of all system components	2.3, 3.3, 4.3
4	updated three layered architecture to Model View Controller is an established design pattern.	Changed naming of three layers. Previously used – Process – which is more applicable to business	System Topology

		applications separation of concerns, previous experience	
5	Separated component into three models for each layer Replaced 4 components by 2 in View Layer, 4 in Controller Defined interfaces for each component	Layered system should have components separated into different subsets	System Components
6	Section 4 was rewritten completely.	Inconsistencies in the old document. Lack of architectural detail	3, 4
7	Added intended audience description	Makes the document more useful and identifies particular parts of document that could be interesting for particular user	1.1
8	Added references to RLCA document dependencies	This document depends on requirements and system definitions specified in other documents	1.3
9	Updated system processes Introduced 2 new use cases	Generation of analysis report is separated into two stages to avoid unnecessary computations when parameters are changed only for one stage	2.3.1
10	Updated Deployment Model	Fixed inconsistency with other diagrams in UML model	3.1.4
11	Updated Implementation Specific Deployment Model	Added specification for the hardware hosting the proposed system	4.1.4
12	Added description of objects used by system	Descriptions were missing	4.1.13
13	Added description for each system process	Descriptions were missing	3.3
14	Rewritten the document	Document did not correspond to the system	All sections

## 2. System Analysis

This section refines the proposed system in terms of its interactions with its users and artifacts it operates on as it was defined in OCD 4 “Proposed system”.

### 2.1 Structure

We describe the actors who interact with the system when it is operational. The proposed system does not have different modes of behavior based on which business or outside actor is interacting with it. Therefore the system will have one generic user and will provide all of its capabilities in one operational mode.

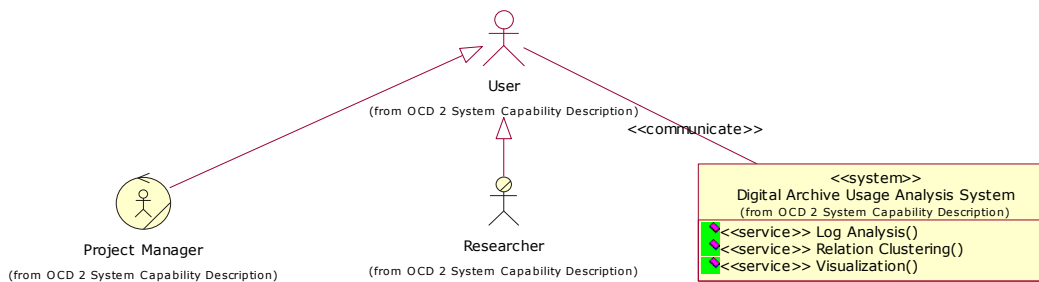


Figure 1 System Static Structure diagram

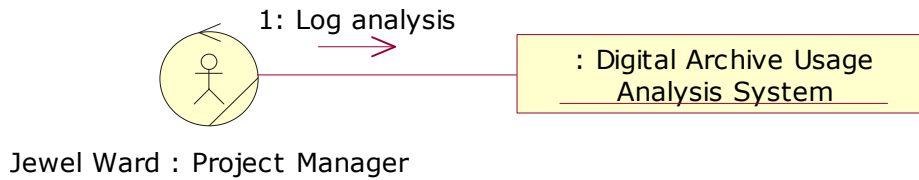


Figure 2 System Collaboration diagram

Figure 2 shows one possible configuration of system usage – the instance of Project Manager invokes one of the system’s capabilities – “Log Analysis”.

#### 2.1.1 System

The proposed system is a tool that allows visualization and analysis of Digital Archive usage data. Provided with input log files the system generates and visualizes digital archive item collection structure that reflects similarity relationships between items that are viewed together most often. In summary the system provides the following major capabilities:

- Log Analysis
- Relation Clustering

- Visualization of analysis results

## 2.1.2 User

The user of the proposed system is a generic actor that is expected to have experience using other software and being familiar with common computer interface concepts such like input forms, action buttons etc. Also familiarity with Digital Archive collections would help the user to utilize the proposed system more efficiently. The user does not have any attributes used by the system and is not required to provide any services.

## 2.2 Artifacts & Information

The following diagram presents artifacts processed and manipulated by the proposed system.

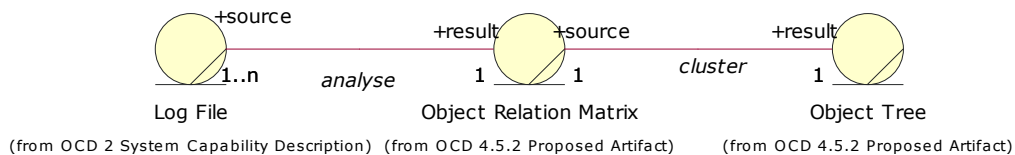


Figure 3 Artifact Model

### 2.2.1 Log File SA-1

The Log File is a text file with each line representing object retrieval from Digital Archive (DA) collection. This artifact is necessary as input data for the system's "Log Analysis" capability.

### 2.2.2 Object Relation Matrix SA-2

Log Analysis capability of the system produces Object Relationship Matrix. It is a representation of object similarities based on co-retrieval events registered in input log files and expressed in matrix form. This artifact is used as input for Object Tree generation.

### 2.2.3 Object Tree SA-3

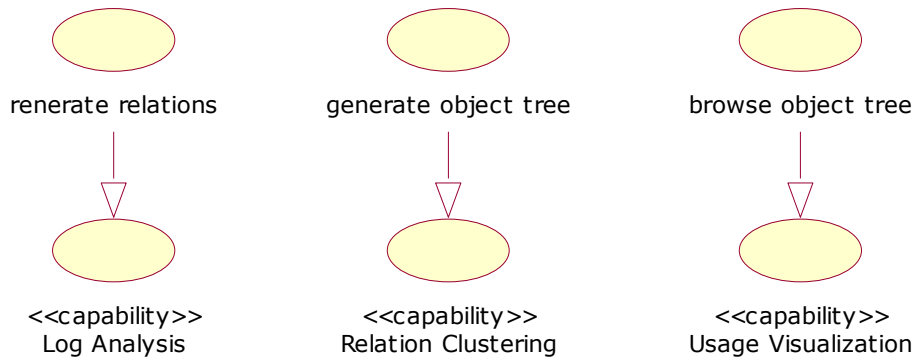
This artifact is a result of applying Relation Cluster capability of the system to input Object Relationship Matrix. This artifact represents a hierarchical tree of Digital Archive objects that expresses high level structure of the DA collection derived based on usage information.

## 2.3 Behavior

This section shows the processes that realize each of the system's capability defined in OCD section 4.3.

### 2.3.1 Processes

Proposed system participates in two processes that implement its three major capabilities as demonstrated in the following diagram.



**Figure 4 System Capability Realization**

The following diagram shows system processes, actors participating and relations among them.

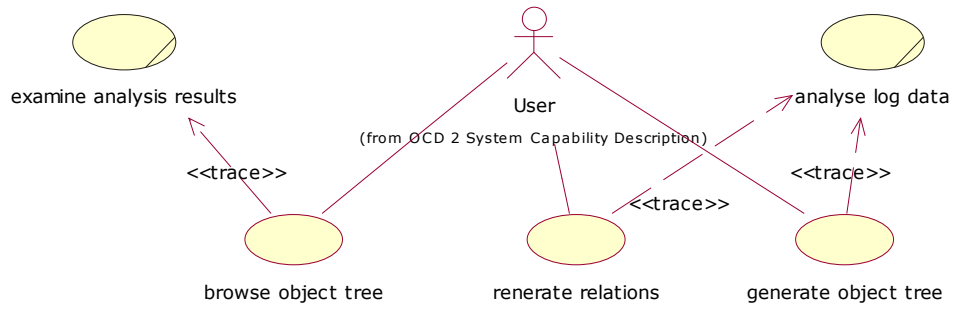


Figure 5 System processes and actors

### 2.3.1.1 Analyze log data UC-01

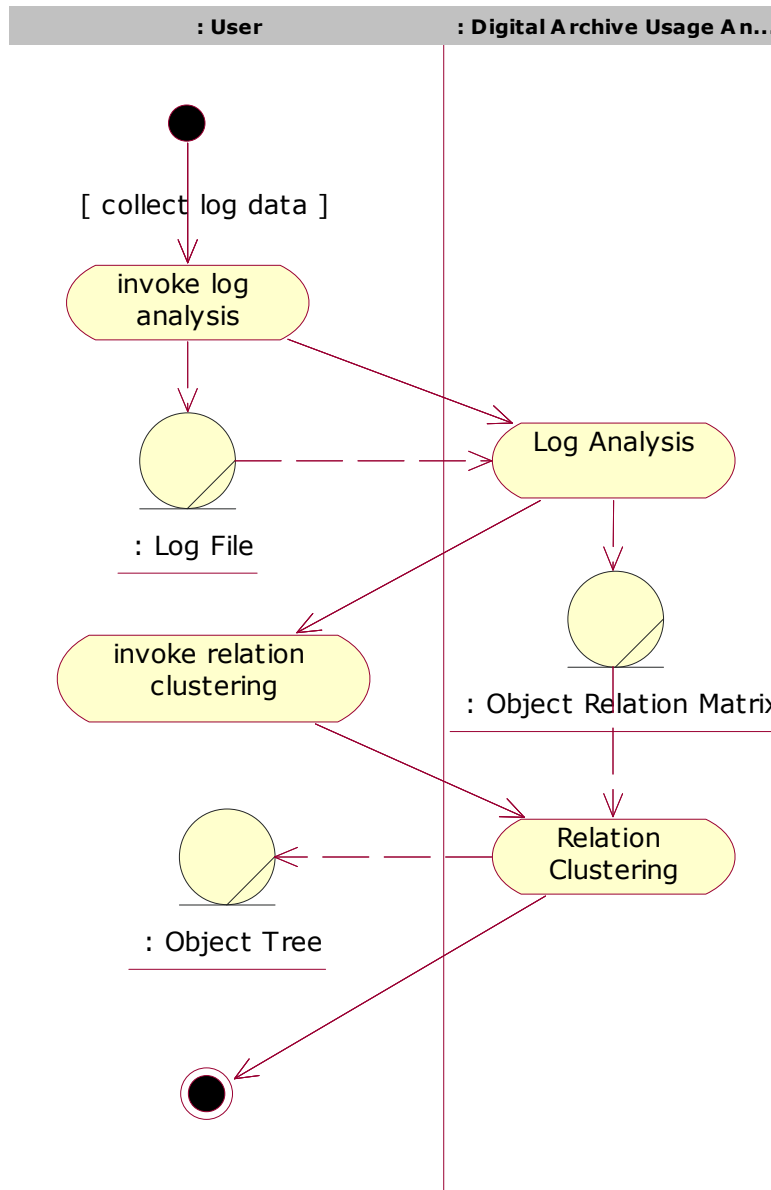
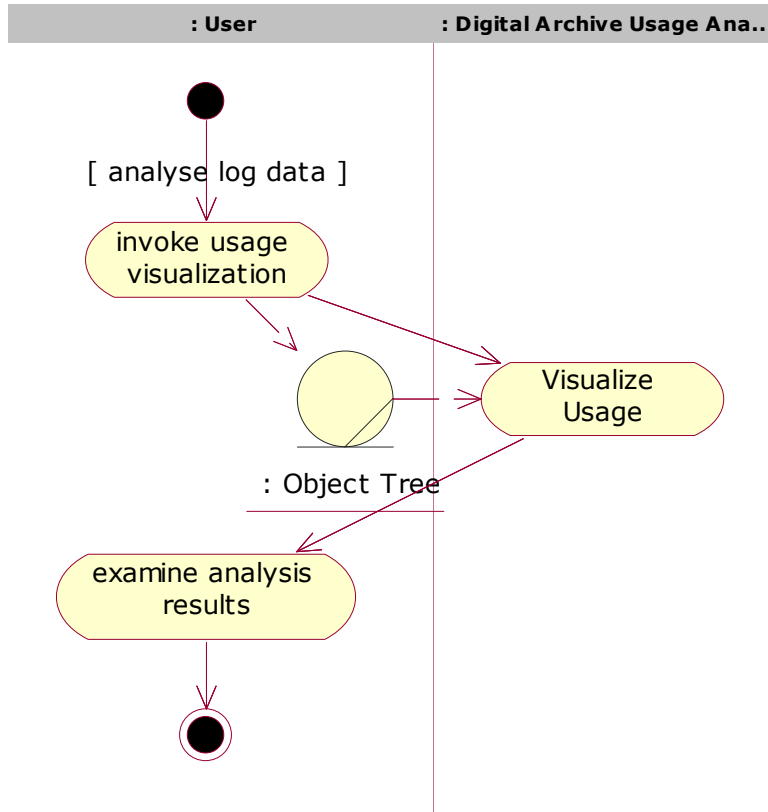


Figure 6 Analyze log data Activity diagram

<b>Identifier</b>	UC-01
<b>Use-Case Name</b>	Analyze log data
<b>Abstract</b>	No
<b>Purpose</b>	Demonstrates how log data is analyzed using proposed system to produce useful information about collection.
<b>Actors</b>	User
<b>Importance</b>	Primary
<b>Capability</b>	C-01, C-02 (OCD 4.3.1)
<b>Requirements</b>	SR-1, SR-2 (SSRD 3.2.1), IR-1 (SSRD 4.1.1)

<b>Risks</b>	None
<b>High-Risk?</b>	No
<b>Architecturally Significant?</b>	No
<b>Development Status</b>	IOC2
<b>Overview</b>	Log data which comes in form of Log File is analyzed in two stages. First system is used to produce object relationship matrix. Clustering is applied to relations to produce hierarchical object tree which represents high level collection structure.
<b>User Interface</b>	Two stages of analysis are done on command line by passing input data and analysis options in the execution command.
<b>Pre-conditions</b>	Source log file is available on the local disk. Proposed system is properly installed.
<b>Post-conditions</b>	Object relation matrix and object tree are generated and stored on the local disk.
<b>Specializes</b>	None
<b>Includes</b>	None
<b>Extends</b>	None
<b>Extension Points</b>	None

**2.3.1.2 Examine analysis results UC-02**



**Figure 7 Examine analysis results activity diagram**

<b>Identifier</b>	UC-02
<b>Use-Case Name</b>	Examine analysis results
<b>Abstract</b>	No
<b>Purpose</b>	Demonstrates how the proposed system is used to visualize results of log analysis
<b>Actors</b>	User
<b>Importance</b>	Primary
<b>Capability</b>	C-01, C-02 (OCD 4.3.1)
<b>Requirements</b>	SR-1, SR-2 (SSRD 3.2.1), IR-1 (SSRD 4.1.1)
<b>Risks</b>	None
<b>High-Risk?</b>	No
<b>Architecturally Significant?</b>	No
<b>Development Status</b>	IOC
<b>Overview</b>	User passes object tree file that was produced by analysis of log data as an input parameter when invoking visualization capability. User is presented with interactive hyperbolic 3d view of object tree. user can browse the tree and change the

	view.
<b>User Interface</b>	Interactive hyperbolic 3d view of the graph. File-system like browsing additional browsing interface. Interface for object search and opening object info in browser window.
<b>Pre-conditions</b>	Object tree file is available on local hard drive.
<b>Post-conditions</b>	Described interface is presented to the user.
<b>Specializes</b>	None
<b>Includes</b>	None
<b>Extends</b>	None
<b>Extension Points</b>	None

**Table 1 Use-case description: Examine analysis results**

## 2.3.2 Modes of Operation

### 2.3.2.1 Operational Mode

The proposed system has only one mode of operation in which it will provide all the capabilities defined in OCD 4.3.

<b>Capability</b>	<b>Processes</b>	<b>Mode Impact</b>
Log Analysis	<ul style="list-style-type: none"> <li>Analyze log data</li> </ul>	N / A
Relation Clustering	<ul style="list-style-type: none"> <li>Analyze log data</li> </ul>	N / A
Visualization	<ul style="list-style-type: none"> <li>Examine analysis results</li> </ul>	N / A

**Table 2 Available Capabilities & Processes in Operational Mode**

## 2.4 L.O.S. Goals

<b>L.O.S requirement</b>	<b>Applies to</b>	<b>How</b>
LR-1: System Dependability - Stable data import/export	UC-01, UC-02	Equally
LR-2 : Usability – User-friendly interface for viewing item relationships and updating data	UC-02	Equally
LR-3: Usability – Maximizing the usability of host resources	UC-02	Equally
LR-4: Performance – Organizing data	UC-01, UC-02	Equally

meaningfully for users		
LR-5: Performance- data of current scale	UC-01, UC-02	Equally

**Table 3 L.O.S. goals**

## 2.5 Rules

None

## 3. Architecture Design & Analysis

The architecture design and analysis provide a high-level architecture for the system and also explain the relationship between the components. This section will describe what are the work units (both hardware and software *components*) of the system and what the components are expected to do.

### 3.1 Structure

#### 3.1.1 Topology

The proposed system is composed of three very simple sub-systems. Each subsystem is run independently and provides certain useful functionality on its own. The sub-systems have common data input/output interfaces and therefore can be used together to produce useful results. The proposed system was decoupled into three sub-systems to make it more flexible and avoid rerunning all analysis stages, when its needed to rerun only one. Because of the simplicity the proposed system is divided into two layers Software and Hardware.

##### 3.1.1.1 Software layer

Contains the three sub-systems: relation generator, tree generator, viewer.

##### 3.1.1.2 Hardware layer

Contains hardware components used to host the proposed sub-systems.

#### 3.1.2 Hardware Classifier Model

This section describes hardware components that are either part of the system or on which this system will run and actors which will interact with the hardware components. The proposed three sub-systems are developed as standalone applications which can be run on any compatible platform. Deployment setup for most common uses of the system would be to run the analysis sub-systems in offline mode on a remote host, and then transfer the resulting data to local host and visualize it using viewer sub-system.

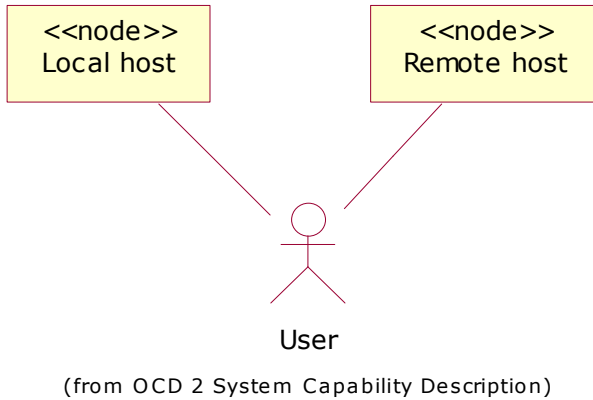


Figure 8 Hardware classifier model

### 3.1.3 Software Classifier Model

The proposed system is composed of three independent sub-systems

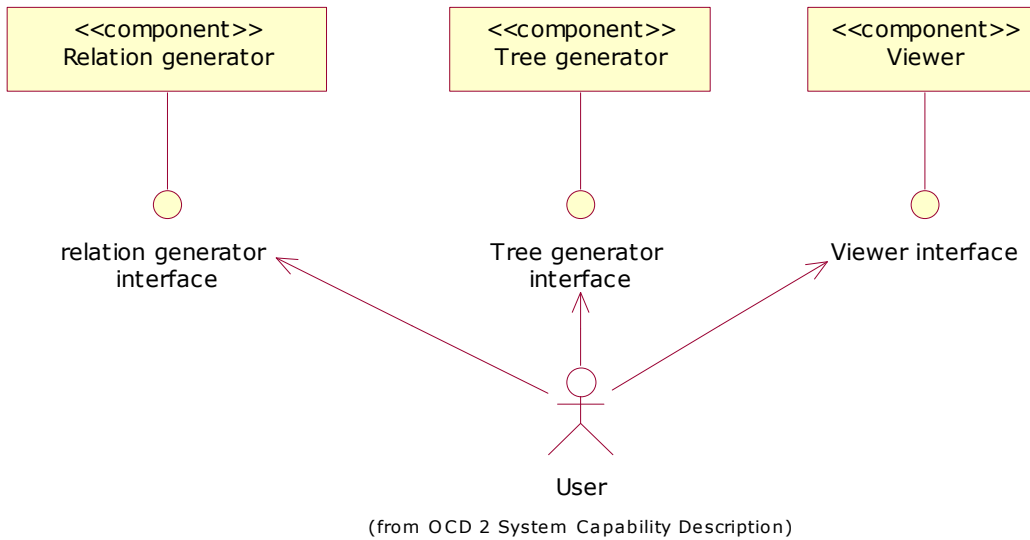
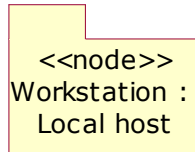


Figure 9 Software classifier model

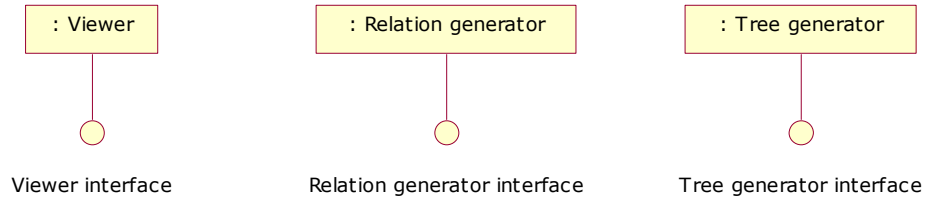
### 3.1.4 Deployment Model

#### 3.1.4.1 Standard configuration

In standard configuration all three sub-systems are deployed on the local user workstation. Each the subsystems is used independently and is passed with input data that is generated by other sub-systems or obtained from logging software.



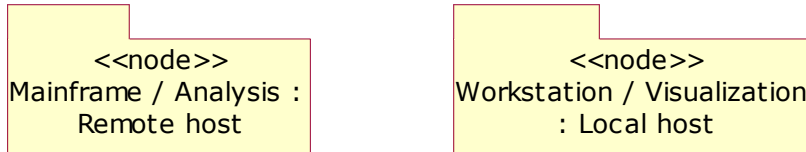
**Figure 10 Standard configuration hardware**



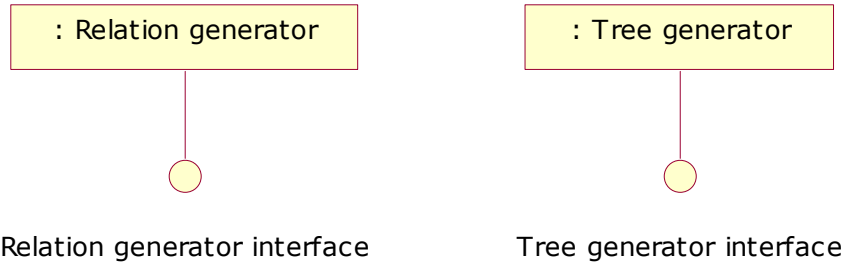
**Figure 11 Standard configuration software**

### 3.1.4.2 Performance configuration

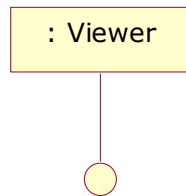
Performance configuration is used to run analysis sub-systems in offline mode on a remote host. It could be used to handle large sets of data and would make local workstation available for other tasks.



**Figure 12 Performance configuration hardware**



**Figure 13 Performance configuration Mainframe software**



Viewer interface

(from Software Component Classifiers)

**Figure 14 Performance configuration Workstation software**

### 3.1.5 Hardware Component Classifiers

#### 3.1.5.1 Local host HCC-01

##### 3.1.5.1.1 Purpose

Local workstation can be used to run all three sub-systems. For most purposes local host has to be used to run Viewer sub-system.

##### 3.1.5.1.2 L.O.S. Goals

None

### 3.1.5.2 Remote host HCC-02

#### 3.1.5.2.1 Purpose

Remote host can be used to run all three sub-systems. For most purposes is used to run relation generator and tree generator in offline mode. Could be used to run viewer sub-system using standard technology for virtual graphical interface to remote workstation such like VPN, X server forwarding etc

#### 3.1.5.2.2 L.O.S. Goals

None

## 3.1.6 Hardware Connector Classifiers

None

## 3.1.7 Software Component Classifiers

### 3.1.7.1 Relation generator SCC-01

#### 3.1.7.1.1 Purpose

This component is used to process log file, analyze contained retrievals and generate object similarity relationships which reflect common usage of objects.

#### 3.1.7.1.2 Interface(s)

Relation generator component has very simple interface.

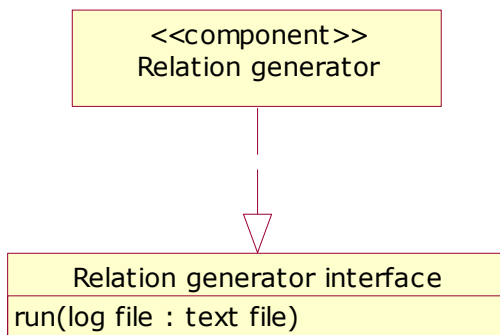


Figure 15 Relation generator interface

#### 3.1.7.1.2.1 Relation generator functionality

Relationship generator parses retrieval events contained in input log file. During the parsing process it detects co-retrieval events for each unique pair of objects during the same browser session. When retrievals with particular session id are not met for predefined time period – the corresponding session is closed and relationships for each unique pair of objects within that session are incremented. After the input file is parsed

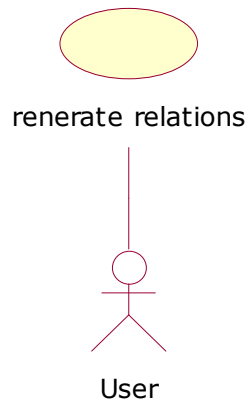
completely the resulting object relationships are stored in predefined format on the local hard disk.

### 3.1.7.1.3 Parameters

Relation generator has one parameter - name of input log file. This parameter is passed when Relation generator is executed.

### 3.1.7.1.4 Behavior

#### 3.1.7.1.4.1 Processes



**Figure 16 Relations generator processes**

3.1.7.1.4.1.1 *Generate relations*

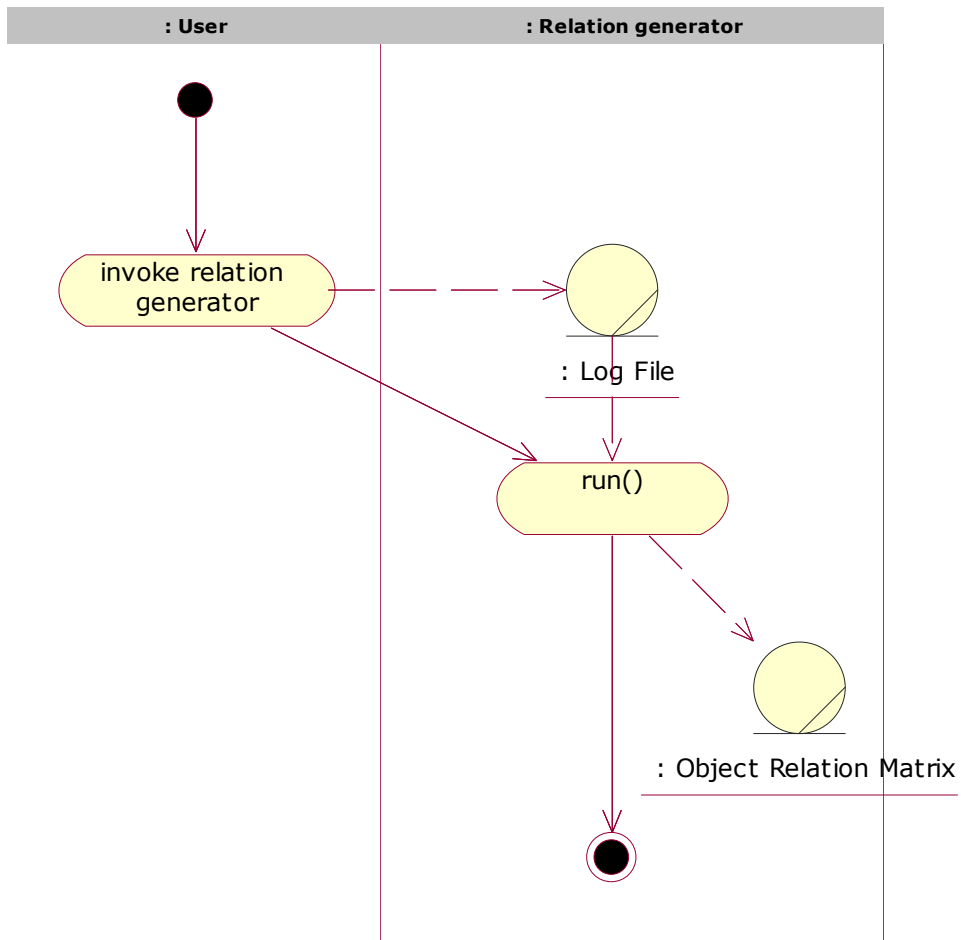


Figure 17 Generate relations Activity diagram

<b>Identifier</b>	PC-1
<b>Use-Case Name</b>	Generate relations
<b>Abstract</b>	No
<b>Purpose</b>	Demonstrates how Object relation matrix is generated
<b>Actors</b>	User
<b>Importance</b>	Primary
<b>Capability</b>	C-01 (OCD 4.3.1)
<b>Requirements</b>	SR-1 (SSRD 3.2.1), IR-1 (SSRD 4.1.1)
<b>Risks</b>	None
<b>High-Risk?</b>	No
<b>Architecturally Significant?</b>	No
<b>Development Status</b>	IOC2

<b>Overview</b>	User invokes Relation generator with input log file as parameter and the tool generates object matrix which is stored to local drive.
<b>User Interface</b>	The tool is invoked from command line.
<b>Pre-conditions</b>	Input log file is available on the local hard drive.
<b>Post-conditions</b>	Object matrix is generated and stored.
<b>Specializes</b>	None
<b>Includes</b>	None
<b>Extends</b>	None
<b>Extension Points</b>	None

**Figure 18 Generate relations use-case description**

#### 3.1.7.1.4.2 Modes of Operation

Operational mode.

#### 3.1.7.1.5 L.O.S. Goals

<b>L.O.S. Requirement:</b>	LR-4: Performance – Organizing data meaningfully for users
<b>Description:</b>	The generated data should be save in a meaning-full organized form to be easily reusable by humans and other applications.
<b>Measurable:</b>	Measured by number of ways data is available to be reused by a person or another applications
<b>Relevant:</b>	SR-5: Graph node statistics OCD 4.4 LG-3: Performance
<b>Specific:</b>	Specifies how meaningfulness of data organization will be measured.

<b>L.O.S. Requirement:</b>	LR-5: Performance – data of current scale
<b>Description:</b>	This component should be able to perform all of its operations given data input of size of up to 300000 objects.
<b>Measurable:</b>	Successful performing operations on input data of the specified size
<b>Relevant:</b>	SR-3: Relationship generation SR-4: Generate collection structure tree SR-5: Graph node statistics SR-6: Visualization OCD 4.4 LG-3: Performance
<b>Specific:</b>	Specifies acceptable scale limit.

### 3.1.7.1.6 Constraints

None

## 3.1.7.2 *Tree generator SCC-02*

### 3.1.7.2.1 Purpose

This component is used to process Object relation matrix and to generate hierarchical object tree that reflects structure of Digital Archive item usage.

### 3.1.7.2.2 Interface

This component has one simple interface.

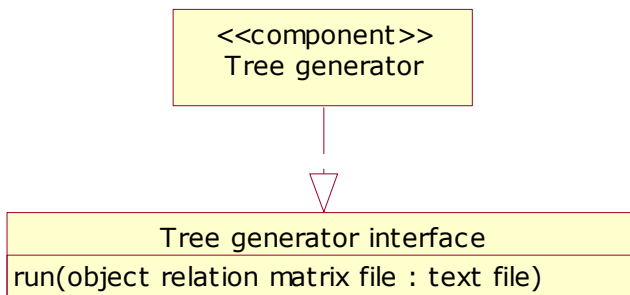


Figure 19 Tree generator interface

### 3.1.7.2.2.1 *Tree generator functionality*

This component is invoked from command line with input object relation matrix file passed as a parameter. When invoked this component parses the input object relation matrix and stores relationship graph in internal representation. After that the tool applies clustering algorithm in hierarchical manner to generate object tree. The resulting tree is saved in predefined format to local drive.

### 3.1.7.2.3 Parameters

This component has one parameter – input object relation file, which is passed when the component is invoked.

### 3.1.7.2.4 Behavior

3.1.7.2.4.1 Processes

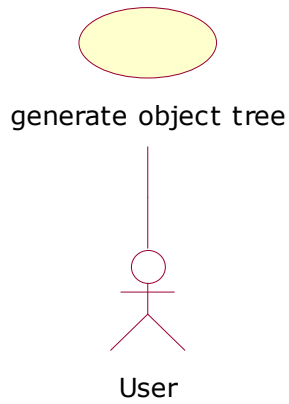
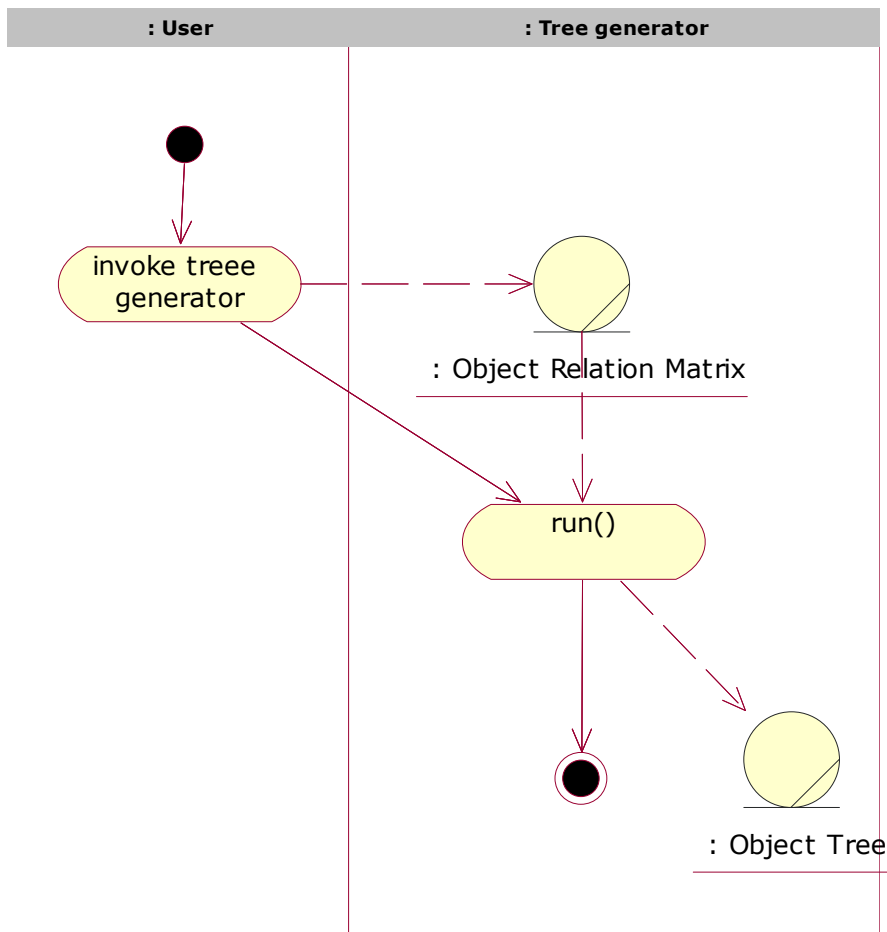


Figure 20 Tree generator processes

3.1.7.2.4.1.1 Generate object tree



**Figure 21 Generate object tree Activity diagram**

<b>Identifier</b>	PC-2
<b>Use-Case Name</b>	Generate relations
<b>Abstract</b>	No
<b>Purpose</b>	Demonstrates how Object tree is generated.
<b>Actors</b>	User
<b>Importance</b>	Primary
<b>Capability</b>	C-02 (OCD 4.3.1)
<b>Requirements</b>	SR-2 (SSRD 3.2.1), IR-2 (SSRD 4.1.1)
<b>Risks</b>	None
<b>High-Risk?</b>	No
<b>Architecturally Significant?</b>	No
<b>Development Status</b>	IOC2
<b>Overview</b>	User invokes Object tree generator with input object relation matrix file as parameter and the tool generates the object tree file which is stored to local drive.
<b>User Interface</b>	The tool is invoked from command line.
<b>Pre-conditions</b>	Input object relation matrix file is available on the local hard drive.
<b>Post-conditions</b>	Object matrix is generated and stored.
<b>Specializes</b>	None
<b>Includes</b>	None
<b>Extends</b>	None
<b>Extension Points</b>	None

**Table 4 Generate object tree description****3.1.7.2.4.2 Modes of Operation**

One mode – operational.

**3.1.7.2.5 L.O.S. Goals**

<b>L.O.S. Requirement:</b>	LR-4: Performance – Organizing data meaningfully for users
<b>Description:</b>	The generated data should be save in a meaning-full organized form to be easily reusable by humans and other applications.
<b>Measurable:</b>	Measured by number of ways data is available to be reused by a person or another applications

<b>Relevant:</b>	SR-5: Graph node statistics OCD 4.4 LG-3: Performance
<b>Specific:</b>	Specifies how meaningfulness of data organization will be measured.

<b>L.O.S. Requirement:</b>	LR-5: Performance – data of current scale
<b>Description:</b>	This component should be able to perform all of its operations given data input of size of up to 300000 objects.
<b>Measurable:</b>	Successful performing operations on input data of the specified size
<b>Relevant:</b>	SR-3: Relationship generation SR-4: Generate collection structure tree SR-5: Graph node statistics SR-6: Visualization OCD 4.4 LG-3: Performance
<b>Specific:</b>	Specifies acceptable scale limit.

### 3.1.7.2.6 Constraints

None

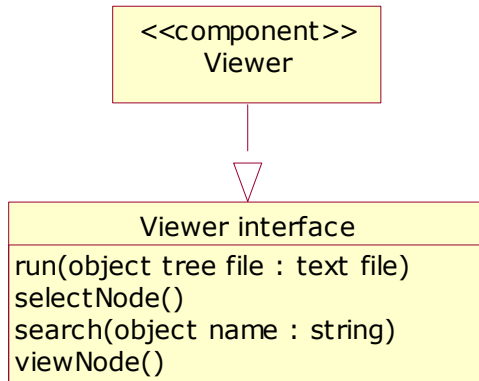
### 3.1.7.3 Viewer SCC-03

#### 3.1.7.3.1 Purpose

This component is used to examine results of analysis done by two previous tools. This component takes object tree file as input and provides interactive visualization of the tree, with browsing and other options.

#### 3.1.7.3.2 Interface(s)

The following operations are available:



**Figure 22 Viewer operations**

### **3.1.7.3.2.1 *run()***

Using this operation the component is invoked from command line and is passed input object tree file for visualization.

### **3.1.7.3.2.2 *selectNode()***

This function is invoked when user selects a tree node in the three dimensional view, or in the browsing list. The viewer panes the 3d view so that the selected node is centered and rotates the view so that children of the node are on the right and parents are on the left.

### **3.1.7.3.2.3 *findNode()***

Given object name entered by the user in the search test widget, the viewer component tries to find the corresponding node in the internal representation. If node with the specified name does not exist viewer does nothing. If the node is found – *selectNode* function is called with the corresponding Node reference.

### **3.1.7.3.2.4 *viewNode()***

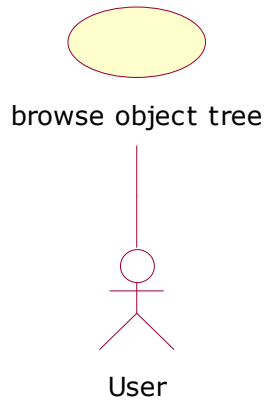
When invoked this function opens an internet browser window with the URL that leads to the web-page with the thumbnail and description of the Digital Archive object that is currently selected.

### **3.1.7.3.3 Parameters**

The only parameter that is passed to this component at invocation time is the input object tree file name.

**3.1.7.3.4 Behavior**

**3.1.7.3.4.1 Processes**



**Figure 23 Viewer processes**

**3.1.7.3.4.1.1 Browse object tree**

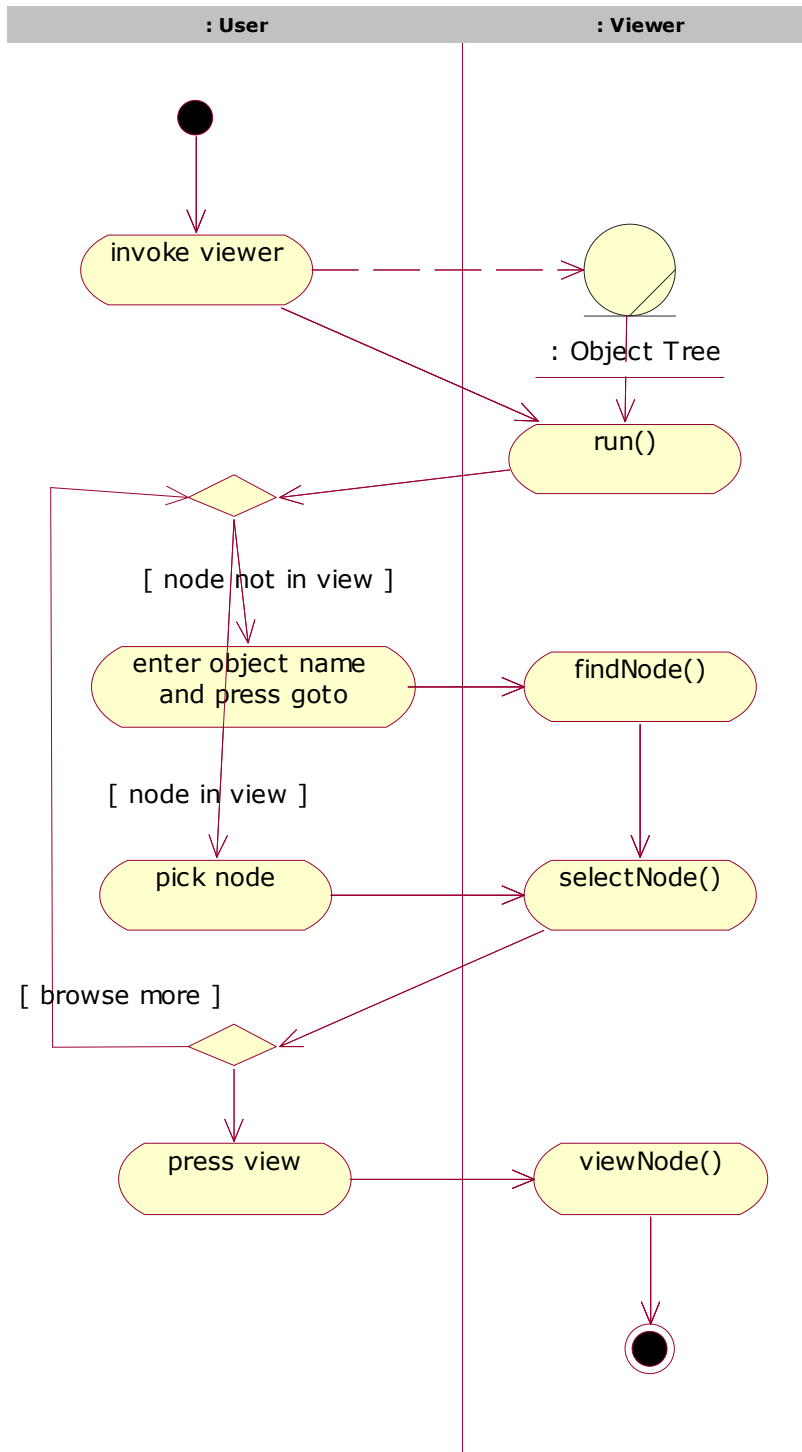


Figure 24 Browse object tree Activity diagram

<b>Identifier</b>	PC-3
<b>Use-Case Name</b>	Browse object tree
<b>Abstract</b>	No
<b>Purpose</b>	Demonstrates how object tree is browsed.
<b>Actors</b>	User
<b>Importance</b>	Primary
<b>Capability</b>	C-03 (OCD 4.3.1)
<b>Requirements</b>	SR-3 (SSRD 3.2.1), IR-3 (SSRD 4.1.1)
<b>Risks</b>	None
<b>High-Risk?</b>	No
<b>Architecturally Significant?</b>	No
<b>Development Status</b>	IOC2
<b>Overview</b>	User invokes Viewer with input object tree file. User is presented with interactive 3d visualization of object tree, where user can pan and rotate the view, select nodes, search for nodes, and open node information in web-browser window.
<b>User Interface</b>	User is presented with interactive 3d hyperbolic view of the graph, where nodes can be selected, and view can be rotated and panned. On the right panel user is presented with a set of selectable scrollable list of object names that represent position of the selected node in the tree hierarchy. User is also presented with search test field and button and view button.
<b>Pre-conditions</b>	Object tree file is available on the local hard drive.
<b>Post-conditions</b>	3d hyperbolic view of the tree is presented.
<b>Specializes</b>	None
<b>Includes</b>	None
<b>Extends</b>	None
<b>Extension Points</b>	None

**Table 5 Browse object tree description**

#### 3.1.7.3.4.2 Modes of Operation

One mode – operational.

#### 3.1.7.3.5 L.O.S. Goals

<b>L.O.S. Requirement:</b>	LR-2 : Usability – User-friendly interface for viewing item relationships and updating data
<b>Description:</b>	The 3d hyperbolic interactive visualization of the Object Trees should be intuitive and informative enough so that user that had short training

	in system usage could use the system with 80% efficiency.
<b>Measurable:</b>	Evaluation by instructed user and estimation efficiency of the user's work.
<b>Relevant:</b>	SR-6: Visualization. OCD 4.4 LS-2: Usability Ensures system usability and therefore satisfaction of the client.
<b>Specific:</b>	Specifies what user efficiency would be sufficient.

<b>L.O.S. Requirement:</b>	LR-3: Usability – Maximizing the host resources
<b>Description:</b>	When doing computationally or IO intensive tasks this component should do them in the background allowing normal operation of regular desktop applications: mail, browser, etc.
<b>Measurable:</b>	Estimated by potential user, during performing computationally intensive tasks – how regular desktop applications are usable.
<b>Relevant:</b>	OCD 4.4 LG-2: Usability Ensures system usability and therefore satisfaction of the client.
<b>Specific:</b>	Specifies what applications should be able to run in parallel.

### 3.1.7.3.6 Constraints

None

## 3.1.8 Software Connector Classifiers

None

## 3.1.9 Hardware Components

### 3.1.9.1 Workstation HC-01

#### 3.1.9.1.1 Purpose

Workstation can be used to run all three subsystems. Usually is used to run Viewer.

#### 3.1.9.1.2 Classifier

HCC-01

#### 3.1.9.1.3 L.O.S.

None

### **3.1.9.2 Mainframe HC-02**

#### **3.1.9.2.1 Purpose**

Mainframe could be used to run all three components of the proposed system, including Viewer – through standard virtual graphical user interface. Usually is used to run Relation generator and Object tree generator.

#### **3.1.9.2.2 Classifier**

HCC-02

#### **3.1.9.2.3 L.O.S.**

None

### **3.1.10 Hardware Connectors**

None

### **3.1.11 Software Components**

#### **3.1.11.1 Relation generator SC-01**

##### **3.1.11.1.1 Purpose**

Process input log file and generate object relation matrix.

##### **3.1.11.1.2 Classifier**

SCC-01

##### **3.1.11.1.3 L.O.S.**

Same as for SCC-01

#### **3.1.11.2 Tree generator SC-02**

##### **3.1.11.2.1 Purpose**

Process input object relation matrix file and generate object tree file.

##### **3.1.11.2.2 Classifier**

SCC-02

##### **3.1.11.2.3 L.O.S.**

Same as for SCC-02

### **3.1.11.3 Viewer SC-03**

#### **3.1.11.3.1 Purpose**

Parse object tree file and present 3d hyperbolic interactive visualization of the tree.

#### **3.1.11.3.2 Classifier**

SCC-03

#### **3.1.11.3.3 L.O.S.**

Same as for SCC-03

## **3.1.12 Software Connectors**

None

## **3.2 Analysis Classes**

This section describes Information Classes that the proposed system operates on.

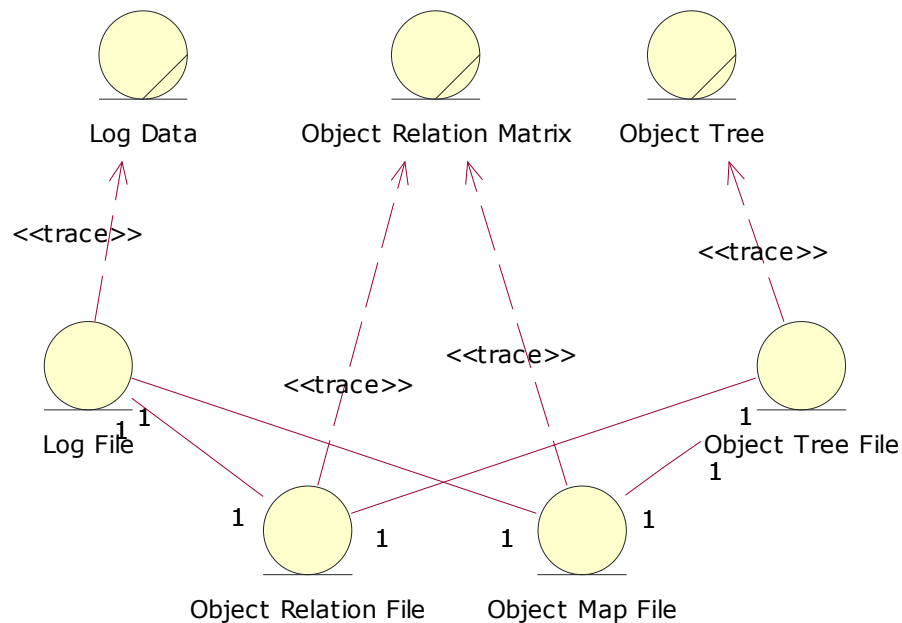


Figure 25 Analysis classes

### 3.2.1 Log File AC-01

The USC-ISD has maintained a log file of usage statistic of the USC digital archive. When users visit the Digital archive web site and retrieve certain images, their activities will be logged, the information such as user's IP, item id and time of retrieval are recorded in the log file. These log files will then be imported by user to their local directory to be analyzed by the system. System Capability C-01 [see OCD 4.3] uses this artifact.

### 3.2.2 Object Relation File AC-02

This class represents relations that have been generated based on available logs. A relationship has the following attributes: relation set id, objectA, objectB, weight. This class is used by the C-02 (OCD 4.3)

### 3.2.3 Object Map File AC-03

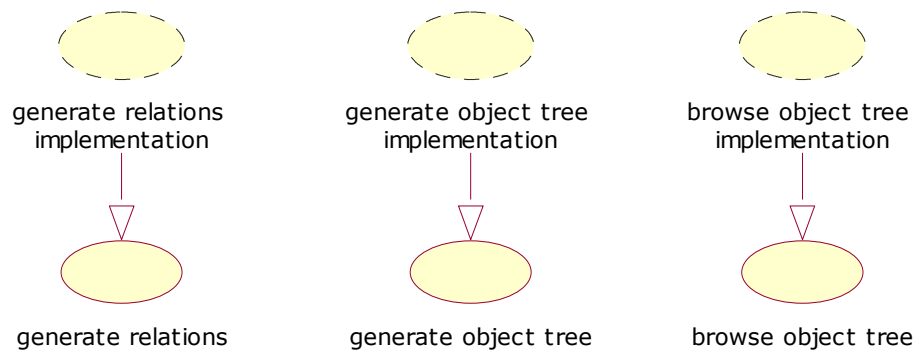
This class represents mapping of object names to unique numeric object ids. This is done to save memory while generating and storing object relations. Object map has the following attributes: object name, object id. This class is used by the C-02 (OCD 4.3)

### 3.2.4 Object Tree File AC-04

This class represents object tree that is created based on a Object Relation File and Object Map File. Object tree represents a set of tree nodes that compose a tree. Viewer component uses this class to construct a 3d hyperbolic view of object tree. This class is used by C-02 and C-03 (OCD 4.3)

## 3.3 Behavior

This section will give more detailed description about how the components in section 3.1.4 work together to implement the processes in section 2.3.



**Figure 26 System process implementation**

### 3.3.1.1 Generate relations implementation

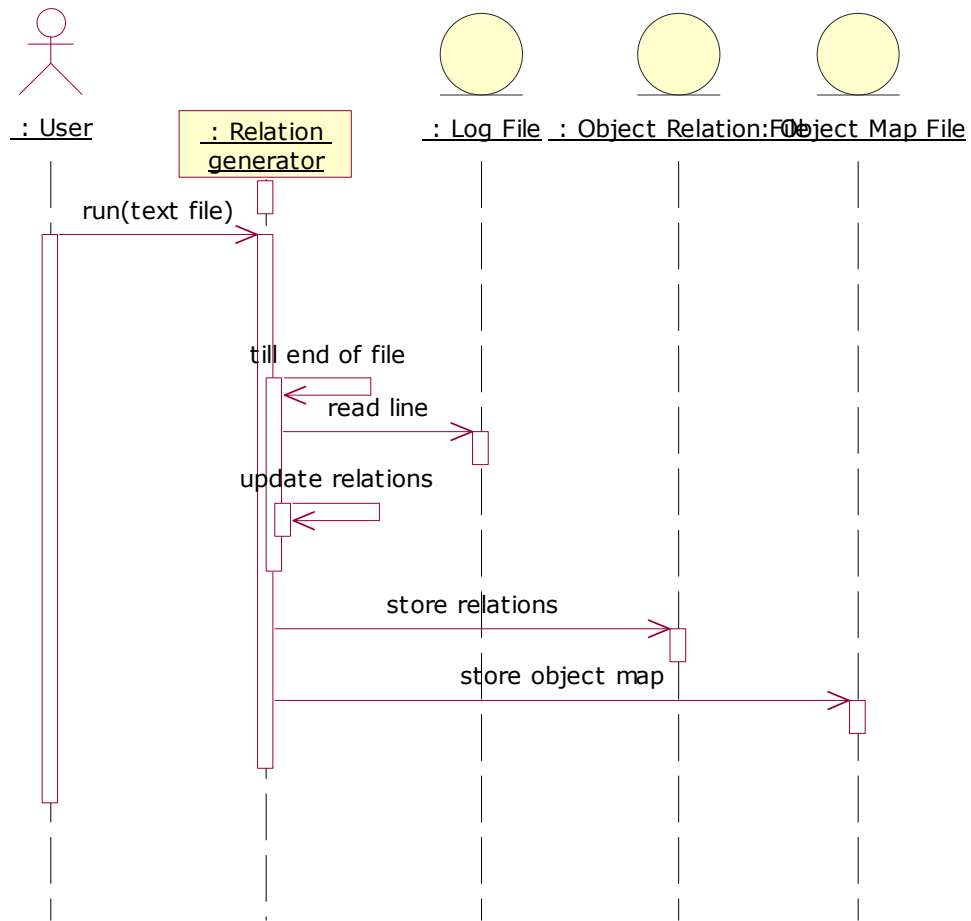


Figure 27 Generate object relations implementation

Table 6 Generate relations implementation

<b>Identifier</b>	UCR-01
<b>Use-Case Name</b>	Generate Relations Implementation
<b>Abstract</b>	No
<b>Purpose</b>	Demonstrates how generation of object relations is implemented.
<b>Actors</b>	User
<b>Importance</b>	Primary
<b>Capability</b>	C-02 (OCD 4.3)
<b>Requirements</b>	SR-3 (SSRD 3.2.1) SR-9 (SSRD 3.2.2) IR-2, 3 (SSRD 4.1.1)
<b>Risks</b>	None
<b>High-Risk?</b>	No
<b>Architecturally</b>	No

<b>Significant?</b>	
<b>Development Status</b>	IOC2
<b>Overview</b>	User invokes relations generator with the input log file. Relation generator proceeds to parse log file and for each parsed retrieval event it updates internal representation of object relations and object map. After end of the file has been reached object relations and object map are stored into corresponding files.
<b>User Interface</b>	Command line tool.
<b>Pre-conditions</b>	Input log file available on the local hard drive.
<b>Post-conditions</b>	Object relation matrix file and Object map file are generated.
<b>Specializes</b>	None
<b>Includes</b>	None
<b>Extends</b>	None
<b>Extension Points</b>	None

Figure 28 Generate relations description

### 3.3.1.2 Generate object tree implementation

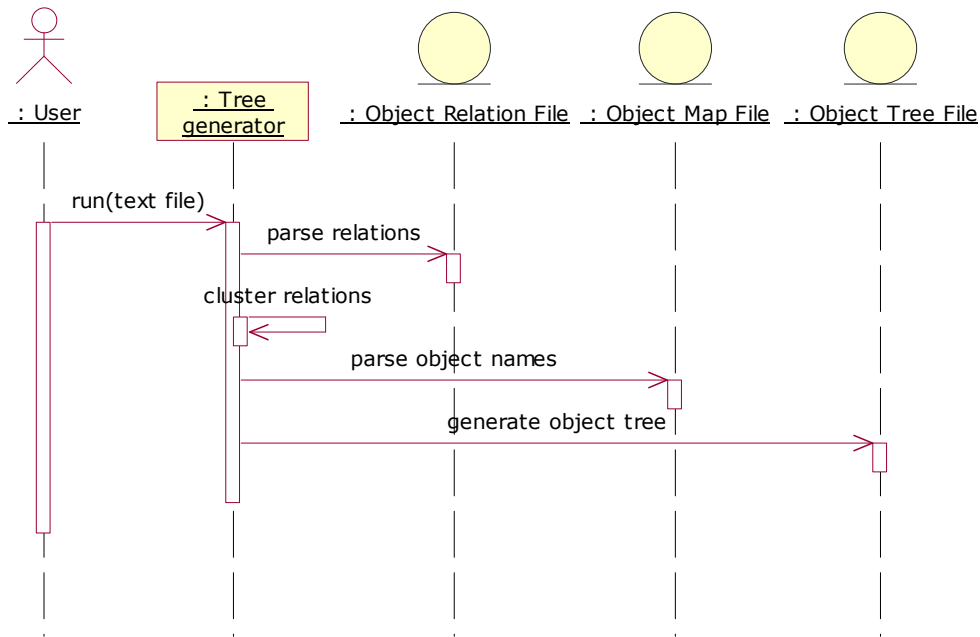
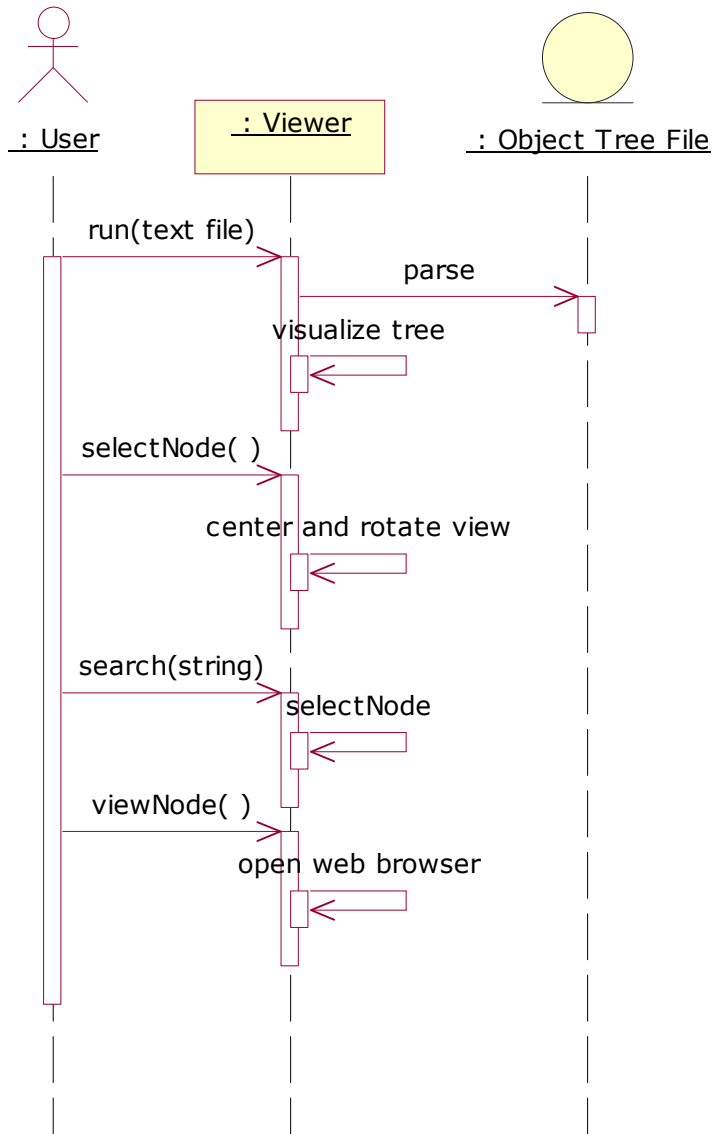


Figure 29 Generate object tree implementation

<b>Identifier</b>	UCR-02
<b>Use-Case Name</b>	Generate Object Tree Implementation
<b>Abstract</b>	No
<b>Purpose</b>	Demonstrates how generation of Object Tree is implemented

<b>Actors</b>	User
<b>Importance</b>	Primary
<b>Capability</b>	C-02 (OCD 4.3)
<b>Requirements</b>	SR-4 (SSRD 3.2.1) IR-2 (SSRD 4.1.1)
<b>Risks</b>	None
<b>High-Risk?</b>	No
<b>Architecturally Significant?</b>	No
<b>Development Status</b>	IOC
<b>Overview</b>	User invokes Tree generator with input Object matrix relations file. Tree generator proceeds to parse relations file and cluster relations in hierarchical manner. After that Tree generator reads object names from Object Map file and generates Object tree, which is stored to local drive.
<b>User Interface</b>	See OCD 5 – prototype
<b>Pre-conditions</b>	Object Matrix file is available from local drive
<b>Post-conditions</b>	Object tree file is stored on local drive
<b>Specializes</b>	None
<b>Includes</b>	None
<b>Extends</b>	None
<b>Extension Points</b>	None

**3.3.1.3 Browse object tree implementation**



**Figure 30 Browse object tree implementation**

<b>Identifier</b>	UCR-03
<b>Use-Case Name</b>	Browse Object Tree Implementation
<b>Abstract</b>	No
<b>Purpose</b>	Demonstrates how interactive browsing of object tree is implemented by the system
<b>Actors</b>	User
<b>Importance</b>	Primary

<b>Capability</b>	C-03 (OCD 4.3)
<b>Requirements</b>	SR-6 (SSRD 3.2.1) SR-10 (SSRD 3.2.2) IR-5 (SSRD 4.1.1)
<b>Risks</b>	None
<b>High-Risk?</b>	No
<b>Architecturally Significant?</b>	No
<b>Development Status</b>	IOC2
<b>Overview</b>	User invokes Viewer with input object tree file. User is presented with interactive 3d visualization of the object tree. User can select node by clicking on it in the 3d view or in the browsing pane on the right. User can search for the node by object name. User can open a browser window with thumbnail and description of the selected object.
<b>User Interface</b>	See OCD 5 – prototype
<b>Pre-conditions</b>	Object tree file is available on local drive.
<b>Post-conditions</b>	Interactive view of the Object Tree.
<b>Specializes</b>	None
<b>Includes</b>	None
<b>Extends</b>	None
<b>Extension Points</b>	None

Table 7 Browse object tree description

### 3.4 L.O.S., Projected

This section describes the elements of the architecture to which each system L.O.S. applies and the projected value of each system L.O.S.

<b>L.O.S Goal</b>	<b>Applies To</b>	<b>How</b>	<b>Projected Value</b>	<b>Evaluation Technique</b>
LR-1 [SSRD 5]: Dependability	SCO-01, SCO -02, SCO -03, SCO -04	Equally	Pass 95% of test cases	Estimation
LR-2 [SSRD 5]: Usability	SCO -04	Equally	Satisfied with the client	Estimation
LR-3 [SSRD 5]: Operability in multitasking environment	SCO -03	Equally	Pass 95% of test cases	Estimation
LR-4[SSRD 5]: Performance on data of current scale	SCO -04	Equally	Able to perform properly with 10% increase.	Estimation

Table 8 L.O.S projected

### 3.5 Architectural Styles, Patterns & Frameworks

The three components of the proposed system are used together in the manner similar to Pipe & Filter design style, with two analysis components being filters and Viewer being consumer – or final sink.

Name	Description	Benefits, Cots & limitations
Pipe & Filter Style	Components of the system process data and cooperate through shared data format.	Ease of system modification and extention.

## 4. Implementation Design

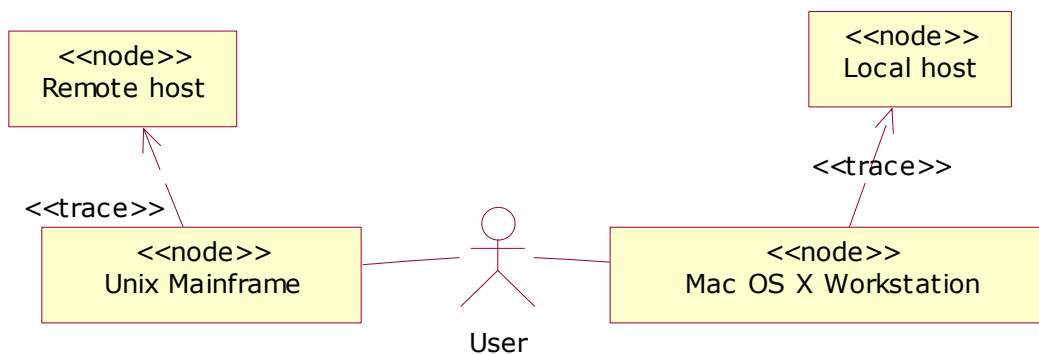
In this section we will present a technology-specific implementation for the system by refining the general architecture defined in SSAD 3.

### 4.1 Structure

#### 4.1.1 Topology

Same as in SSAD 3.1.1

#### 4.1.2 Hardware Classifier Model



**Figure 31 Hardware Classifier Model**

This figure displays implementation specific hardware components that will be used by the system.

### 4.1.3 Software Classifier Model

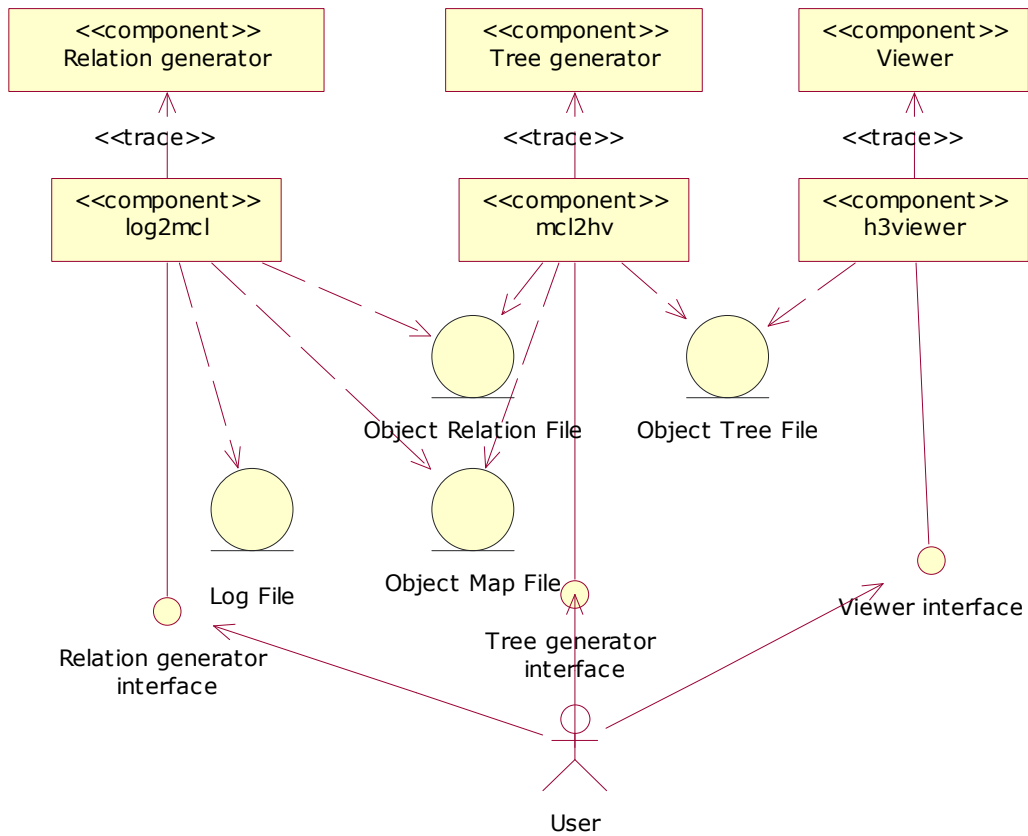


Figure 32 Software classifier model

### 4.1.4 Deployment Model

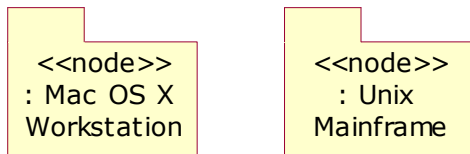
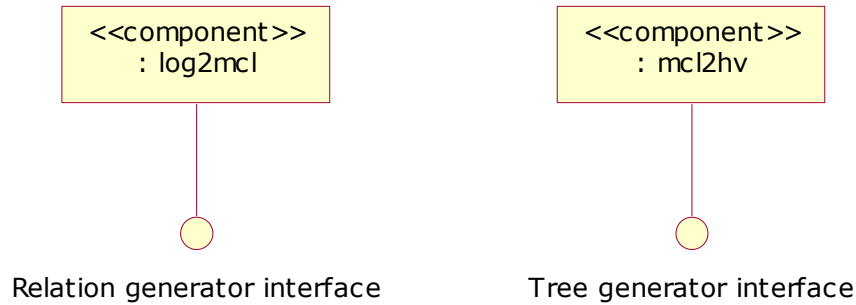
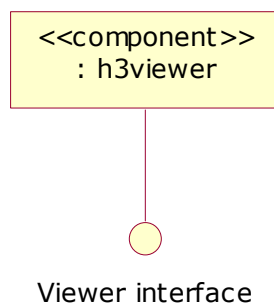


Figure 33 Hardware components used



**Figure 34 Software components on Unix Mainframe**



**Figure 35 Software components on Mac OS X Workstation**

## 4.1.5 Hardware Component Classifiers

### 4.1.5.1 Mac OS X Workstation IHCC-01

#### 4.1.5.1.1 Purpose

Used to host h3viewer component to urn visualization and browsing.

### 4.1.5.2 UNIX Mainframe IHCC-02

#### 4.1.5.2.1 Purpose

Used to host log3mcl and mcl2hv to run log analysis.

## 4.1.6 Hardware Connector Classifiers

None

## 4.1.7 Software Component Classifiers

### **4.1.7.1 log2mcl ISCC-01**

#### **4.1.7.1.1 Purpose**

See 3.1.7.1.1

#### **4.1.7.1.2 Interface(s)**

See 3.1.7.1.2

#### **4.1.7.1.3 Parameters**

See 3.1.7.1.3

#### **4.1.7.1.4 Behavior**

See 3.1.7.1.4

##### **4.1.7.1.4.1 Modes of Operation**

One mode - operational.

#### **4.1.7.1.5 L.O.S. Goals**

See 3.1.7.1.5

#### **4.1.7.1.6 Constraints**

Should be able to parse log files of the following format: line number, ip, date, time, object name, session id. Should store object relations in the format defined here: <http://micans.org/mcl/man/mcxio.html>.

#### **4.1.7.1.7 Internal Architecture**

This component is built using functionality of STL classes and defined two classes of its own: log2hvMain and Session. STL classes – stlMap and stlVector are used for data storage that allows dynamic sizes searches etc.

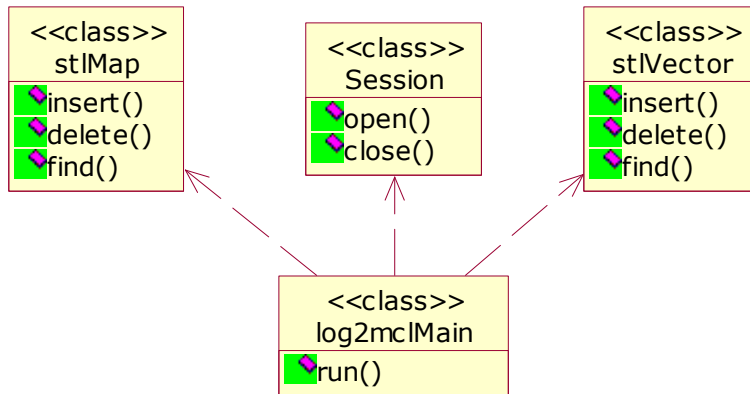


Figure 36 log2mcl Internal Architecture

## 4.1.7.2 mcl2hv ISCC-02

### 4.1.7.2.1 Purpose

See 3.1.7.2.1

### 4.1.7.2.2 Interface(s)

See 3.1.7.2.2

### 4.1.7.2.3 Parameters

See 3.1.7.2.3

### 4.1.7.2.4 Behavior

See 3.1.7.2.4

#### 4.1.7.2.4.1 Modes of Operation

One mode - operational.

### 4.1.7.2.5 L.O.S. Goals

See 3.1.7.2.5

### 4.1.7.2.6 Constraints

Should be able to parse object relations from a matrix file of format defined here: <http://micans.org/mcl/man/mcxio.html>. Should be able to parse mapping of object names to numeric ids from a text file which is a list of object names. Should save object tree in a

format defined here:

[http://graphics.stanford.edu/~munzner/h3/HypView.html#INPUT\\_FILE\\_FORMAT](http://graphics.stanford.edu/~munzner/h3/HypView.html#INPUT_FILE_FORMAT)

#### 4.1.7.2.7 Internal Architecture

This component uses modified clustering functionality implemented by the mcl COTS. Also this component uses data structure that represents relationship matrix and associated helper functions.

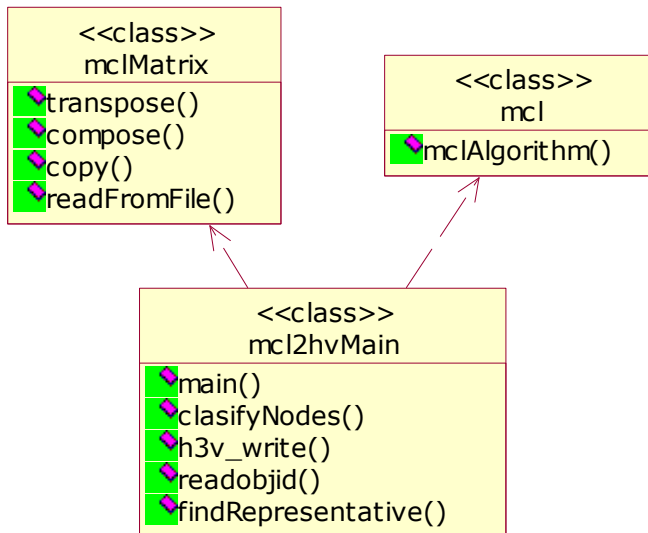


Figure 37 mcl2hv Internal Architecture

### 4.1.7.3 h3viewer ISCC-03

#### 4.1.7.3.1 Purpose

See 3.1.7.3.1

#### 4.1.7.3.2 Interface(s)

See 3.1.7.3.2

#### 4.1.7.3.3 Parameters

See 3.1.7.3.3

#### 4.1.7.3.4 Behavior

See 3.1.7.3.4

#### 4.1.7.3.4.1 Modes of Operation

One mode - operational.

#### 4.1.7.3.5 L.O.S. Goals

See 3.1.7.3.5

#### 4.1.7.3.6 Constraints

Should be able to parse and visualize object tree representation in the following format:  
[http://graphics.stanford.edu/~munzner/h3/HypView.html#INPUT\\_FILE\\_FORMAT](http://graphics.stanford.edu/~munzner/h3/HypView.html#INPUT_FILE_FORMAT).

#### 4.1.7.3.7 Internal Architecture

This component uses H3Viewer class – which is refactored and extended version of example viewer that is provided with open source distribution of h3viewer COTS.

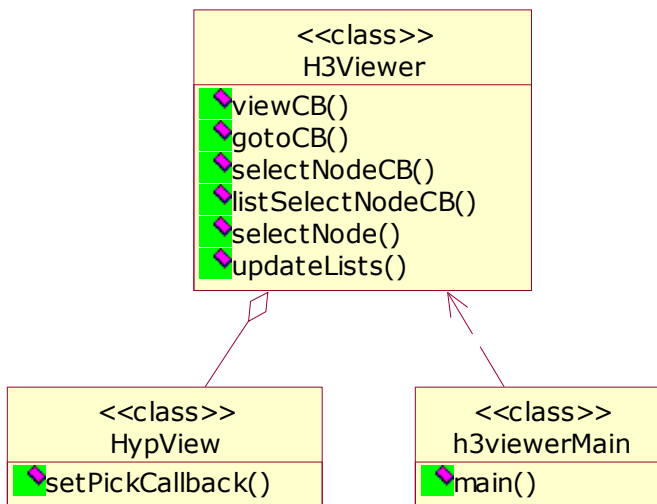


Figure 38 h3viewer Internal Architecture

### 4.1.8 Software Connector Classifiers

None

### 4.1.9 Hardware Components

#### 4.1.9.1 Mac OS X Workstation

##### 4.1.9.1.1 Purpose

See 4.1.5.1.1

**4.1.9.1.2 Classifier**

IHCC-01

**4.1.9.1.3 L.O.S.**

None

**4.1.9.2 UNIX Mainframe**

**4.1.9.2.1 Purpose**

See 4.1.5.2.1

**4.1.9.2.2 Classifier**

IHCC-02

**4.1.9.2.3 L.O.S.**

None

**4.1.10 Hardware Connectors**

None

**4.1.11 Software Components**

**4.1.11.1 log2mcl ISC-01**

**4.1.11.1.1 Purpose**

See 4.1.7.1.1

**4.1.11.1.2 Classifier**

ISCC-01

**4.1.11.1.3 L.O.S.**

See 4.1.7.1.5

**4.1.11.2 mcl2hv ISC-02**

**4.1.11.2.1 Purpose**

See 4.1.7.2.1

**4.1.11.2.2 Classifier**

ISCC-02

**4.1.11.2.3 L.O.S.**

See 4.1.7.2.5

**4.1.11.3 h3viewer ISC-03****4.1.11.3.1 Purpose**

See 4.1.7.3.1

**4.1.11.3.2 Classifier**

ISCC-03

**4.1.11.3.3 L.O.S.**

See 4.1.7.3.5

**4.1.12 Software Connectors**

None

**4.1.13 Implementation Classes****4.1.13.1 log2mclMain****4.1.13.1.1 Purpose**

Implements main functionality of the log2mcl component

**4.1.13.1.2 Defined In: ISC-01****4.1.13.1.3 Interface**

Has one interface with the following operations

<<class>> log2mclMain
run()

**4.1.13.1.4 Parameters**

None

**4.1.13.1.5 Attributes**

None

**4.1.13.1.6 Operations****4.1.13.1.6.1 main OP-1.1**

Identifier: OP-01

Operation name: main

Parameters: Path to the input log file

Result: None

Purpose: Implements core log analysis functionality

Pre-conditions: Input log file is available on the local drive

Post-conditions: Object relation matrix and object map are generated and saved

Visibility: public

Abstract: No

Method: Uses Market Basket analysis technique applied to unique objects being accessed within the same session. Session is considered to be a basket and all items within basket are assumed to be related and relations are incremented correspondingly.

**4.1.13.1.7 State Behavior**

None

**4.1.13.1.8 L.O.S.**

See 4.1.7.1.5

**4.1.13.1.9 Constraints**

None

**4.1.13.2 Session****4.1.13.2.1 Purpose**

Represents a set of retrievals within the same web-browser session. Is used as a data structure to store parsed retrieval information and to update item relations when closed.

**4.1.13.2.2 Defined In: ISC-01****4.1.13.2.3 Interface(s)**

Has one interface with the following parameters

<<class>> Session
open()
close()

**4.1.13.2.4 Parameters**

None

**4.1.13.2.5 Attributes**

None

**4.1.13.2.6 Operations*****4.1.13.2.6.1 open OP-2.1***

Identifier: OP-2.1

Operation name: open

Parameters: Session id

Result: Reference to initialized session

Purpose: Allocates memory for new Session data structure

Pre-conditions: Session key is available, more memory is available.

Post-conditions: Session is initialized and reference is returned

Visibility: public

Abstract: No

Method: Uses standard memory allocation API.

***4.1.13.2.6.2 close OP-2.2***

Identifier: OP-2.2

Operation name: close

Parameters: None

Result: None

Purpose: Updates relationships for each unique pair of objects accessed within this session. Deallocates memory.

Pre-conditions: Session was properly initialized

Post-conditions: Corresponding relations are updated. Memory is freed.

Visibility: public

Abstract: No

Method: Increments relationships by one. Standard memory deallocation.

**4.1.13.2.7 State Behavior**

None

**4.1.13.2.8 L.O.S.**

See 4.1.7.1.5

**4.1.13.2.9 Constraints**

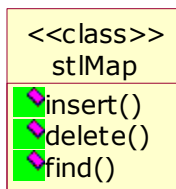
None

**4.1.13.3 stIMap****4.1.13.3.1 Purpose**

Used as a data structure for storing and searching links to currently opened sessions.

**4.1.13.3.2 Defined In: ISC-01****4.1.13.3.3 Interface(s)**

Has one interface with the following operations

**4.1.13.3.4 Parameters**

None

**4.1.13.3.5 Attributes**

None

**4.1.13.3.6 Operations****4.1.13.3.6.1 Insert 3.1**

Identifier: OP-3.1

Operation name: close

Parameters: Object key, Object data.

Result: None

Purpose: Insert object data and associates it with object key for future retrievals.

Pre-conditions: Object key is no null

Post-conditions: Corresponding association is established

Abstract: No

Method: Standard map implementation

**4.1.13.3.6.2 delete OP 3.2**

Identifier: OP-3.2

Operation name: delete

Parameters: Object key

Result: None

Purpose: Removes object data associated with specified object key.

Pre-conditions: Object key is no null

Post-conditions: Corresponding association is removed, memory freed

Abstract: No

Method: Standard map implementation

#### **4.1.13.3.6.3 find OP-3.3**

Identifier: OP-3.3

Operation name: find

Parameters: Object key

Result: Object data associated with the key or null

Purpose: Finds object data associated with specified object key and returns it.

Pre-conditions: Map was properly initialized

Post-conditions: Corresponding object data or null is returned.

Abstract: No

Method: Standard map implementation

#### **4.1.13.3.7 State Behavior**

None

#### **4.1.13.3.8 L.O.S.**

See 4.1.7.1.5

#### **4.1.13.3.9 Constraints**

None

### **4.1.13.4 stlSet**

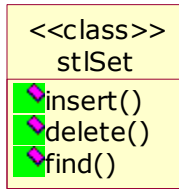
#### **4.1.13.4.1 Purpose**

Used to store a set of unique object names that have non-zero relations among them. It is used later to generate object name map.

#### **4.1.13.4.2 Defined In: ISC-01**

#### **4.1.13.4.3 Interface**

Has one interface with the following operations



#### 4.1.13.4.4 Parameters

None

#### 4.1.13.4.5 Attributes

None

#### 4.1.13.4.6 Operations

##### 4.1.13.4.6.1 *insert OP-4.1*

Identifier: OP-4.1

Operation name: insert

Parameters: Object

Result: Object is inserted, if already exists – does nothing

Purpose: Inserts preserving uniqueness of stored objects.

Pre-conditions: Object is not null

Post-conditions: Object inserted if does not exists.

Abstract: No

Method: Standard set implementation

##### 4.1.13.4.6.2 *remove OP-4.2*

Identifier: OP-4.2

Operation name: remove

Parameters: Object

Result: Object is removed if exists – otherwise does nothing

Purpose: Allows removal of object.

Pre-conditions: Object is not null

Post-conditions: Object is removed.

Abstract: No

Method: Standard set implementation

##### 4.1.13.4.6.3 *find OP-4.3*

Identifier: OP-4.3

Operation name: get

Parameters: Object

Result: true if object exists – otherwise returns false

Purpose: Allows checking object existence

Pre-conditions: Object is not null  
 Post-conditions: True or false returned  
 Abstract: No  
 Method: Standard set implementation

#### 4.1.13.4.7 State Behavior

None

#### 4.1.13.4.8 L.O.S.

See 4.1.7.1.5

#### 4.1.13.4.9 Constraints

None

### 4.1.13.5 mcl2hvMain

#### 4.1.13.5.1 Purpose

Implements core functionality of tree generation.

#### 4.1.13.5.2 Defined In: ISC-02

#### 4.1.13.5.3 Interface(s)

Has one interface with the following operations

<<class>> mcl2hvMain
main() classifyNodes() h3v_write() readobjid() findRepresentative()

#### 4.1.13.5.4 Parameters

None

#### 4.1.13.5.5 Attributes

None

#### 4.1.13.5.6 Operations

**4.1.13.5.6.1 main OP-5.1**

Identifier: OP-5.1

Operation name: main

Parameters: Input Object relation matrix file

Result: None

Purpose: Implements main tree generation functionality. Instantiates other objects and invokes their operations.

Pre-conditions: Object relation matrix file is available

Post-conditions: Object tree file is generated and stored.

Abstract: No

Method: Uses hierarchical clustering and centrality analysis on object relation graph to produce the tree.

**4.1.13.5.6.2 classifyNodes OP-5.2**

Identifier: OP-5.2

Operation name: classifyNodes

Parameters: graph mclMatrix

Result: None

Purpose: Given a mclMatrix that represents a graph of object relations, updates the internal tree representation so that given graph is split several sub-graphs with nodes belonging to only one DA collection..

Pre-conditions: Object relation matrix was successfully parsed

Post-conditions: Bottom level nodes are grouped by collection.

Abstract: No

Method: Uses hierarchical clustering and centrality analysis.

**4.1.13.5.6.3 h3v\_write OP-5.3**

Identifier: OP-5.3

Operation name: h3v\_write

Parameters: graph mclMatrix

Result: None

Purpose: Given that internal representation of object tree is complete and that object name map has been parsed successfully, writes object tree representation into a file in h3viewer format.

Pre-conditions: Hierarchical clustering was done. Object names were parsed.

Post-conditions: Representation of object tree is stored in H3Viewer format.

Abstract: No

Method: Uses h3viewer format definition and Depth First Search.

**4.1.13.5.6.4 readobjid OP-5.4**

Identifier: OP-5.5

Operation name: readobjid

Parameters: path to the object name map

Result: None

Purpose: Parses object name map and stores it internally for future lookups.

Pre-conditions: Object name map is available on the local drive.

Post-conditions: Object name map is parsed and stored internally.

Abstract: No

Method: Uses h3viewer format definition and Depth First Search.

#### **4.1.13.5.6.5 findRepresentative OP-5.5**

Identifier: OP-5.5

Operation name: findRepresentative

Parameters: mclMatrix graph representation

Result: None

Purpose: Given mclMatrix representation of object relation graph identifies most central item in the graph and returns its id

Pre-conditions: Object relation matrix was successfully parsed

Post-conditions: node of the representative is returned

Abstract: No

Method: Uses centrality analysis

#### **4.1.13.5.7 State Behavior**

None.

#### **4.1.13.5.8 L.O.S.**

See 2.1.7.2.5

#### **4.1.13.5.9 Constraints**

None

### **4.1.13.6 mclMatrix**

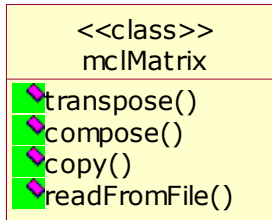
#### **4.1.13.6.1 Purpose**

Is used as data structure for internal representation of object relation graph and for manipulation on that data.

#### **4.1.13.6.2 Defined In: ISC-02**

#### **4.1.13.6.3 Interface**

Has one interface with the following operations



#### 4.1.13.6.4 Parameters

None

#### 4.1.13.6.5 Attributes

None

#### 4.1.13.6.6 Operations

##### 4.1.13.6.6.1 *transpose*

Changes internal representation so that columns of the matrix become rows and vice versa.

##### 4.1.13.6.6.2 *compose*

Given reference to another matrix with corresponding dimensions computes product of two matrixes and returns matrix result.

##### 4.1.13.6.6.3 *copy*

Returns a copy of the current matrix.

##### 4.1.13.6.6.4 *readFromFile*

Parses specified file according to mcl format and creates internal representation of corresponding nodes and weights.

#### 4.1.13.6.7 State Behavior

None

**4.1.13.6.8 L.O.S.**

See 4.1.7.2.5

**4.1.13.6.9 Constraints**

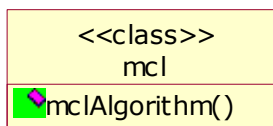
None

**4.1.13.7 mcl****4.1.13.7.1 Purpose**

Implements graph clustering functionality

**4.1.13.7.2 Defined In: ISC-02****4.1.13.7.3 Interface**

Has one interface with the following operations

**4.1.13.7.4 Parameters**

None

**4.1.13.7.5 Attributes**

None

**4.1.13.7.6 Operations****4.1.13.7.6.1 mclAlgorithm**

Given input symmetric mclMatrix which represents acyclic bi-directional graph, groups nodes according to strength of their connection and returns another mclMatrix which represents identified clusters and membership of node in them.

**4.1.13.7.7 State Behavior**

None

**4.1.13.7.8 L.O.S.**

See 4.1.7.2.5

**4.1.13.7.9 Constraints**

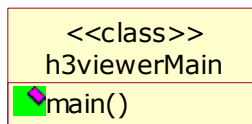
None

**4.1.13.8 h3viewerMain****4.1.13.8.1 Purpose**

Launches the application. Makes necessary initialization.

**4.1.13.8.2 Defined In: ISC-03****4.1.13.8.3 Interface**

Has one interface with the following operations

**4.1.13.8.4 Parameters**

None

**4.1.13.8.5 Attributes**

None

**4.1.13.8.6 Operations**

**4.1.13.8.6.1 main**

Instantiates other objects and invokes their functionality. Launches the GUI event loop.

**4.1.13.8.7 State Behavior**

None

**4.1.13.8.8 L.O.S.**

See 4.1.7.3.5

**4.1.13.8.9 Constraints**

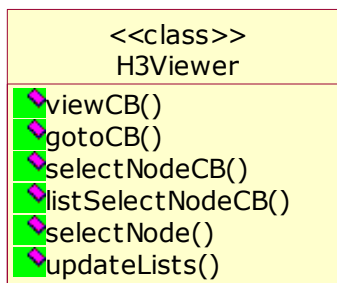
None.

**4.1.13.9 H3viewer****4.1.13.9.1 Purpose**

Implements extension of user interface to stock h3viewer COTS.

**4.1.13.9.2 Defined In: ISC-03****4.1.13.9.3 Interface**

Has one interface with the following operations

**4.1.13.9.4 Parameters**

None

#### **4.1.13.9.5 Attributes**

None

#### **4.1.13.9.6 Operations**

##### ***4.1.13.9.6.1 viewCB***

Handles event of user pressing View button – launches a browser window with corresponding URL.

##### ***4.1.13.9.6.2 gotoCB***

Handles event of user pressing enter in the search text field. Tries to locate the node with specified name in internal representation. If successful selectNode and updateLists are called with corresponding node id, otherwise nothing is done.

##### ***4.1.13.9.6.3 selectNodeCB***

Handles event of user selecting a node in the 3d hyperbolic view. Calls selectNode and updateLists with corresponding node id.

##### ***4.1.13.9.6.4 listSelectNodeCB***

Handles event of user selecting the node in the right side browsing panel. Calls selectNode and updateLists with corresponding node id.

##### ***4.1.13.9.6.5 selectNode***

If passed id is neither node nor edge – does nothing. If passed with id of an edge – calls gotoPickPoint – to center the view on the point where mouse was clicked. If passed id is node – calls gotoNode and setSelected with the corresponding node id.

##### ***4.1.13.9.6.6 updateLists***

Updates right browsing pane such that: list next to the last on the bottom – represents level of the tree where currently selected node is located and thus contains all the siblings of the node and has the corresponding node current selection. List last on the bottom contains all the children on the currently selected node and has no node selected. All levels above represent nodes at each level of the tree above the one of focus node and each list have the ancestor of the node selected.

#### **4.1.13.9.7 State Behavior**

None

**4.1.13.9.8 L.O.S.**

See 4.1.7.3.5

**4.1.13.9.9 Constraints**

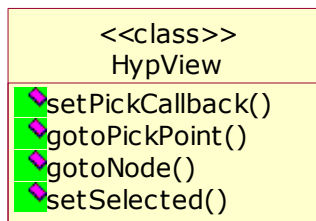
None

**4.1.13.10 HypView****4.1.13.10.1 Purpose**

Implements 3d hyperbolic interactive visualization of object relation tree.

**4.1.13.10.2 Defined In: ISC-03****4.1.13.10.3 Interface**

Has one interface with the following parameters

**4.1.13.10.4 Parameters**

None

**4.1.13.10.5 Attributes**

None

**4.1.13.10.6 Operations****4.1.13.10.6.1 setPickCallback**

Sets handler of the event for mouse clicks in the visualization area

**4.1.13.10.6.2 gotoPickPoint**

Orients the view that that selected edge is the center.

**4.1.13.10.6.3 gotoNode**

Rotates and pans the view such that the selected node is the center, all its children are on the right and all its ancestors are on the left.

**4.1.13.10.6.4 setSelected**

Highlights the selected node with the corresponding color.

**4.1.13.10.7 State Behavior**

None

**4.1.13.10.8 L.O.S.**

See 4.1.7.3.5

**4.1.13.10.9 Constraints**

None

**4.1.14 Objects****4.1.14.1 Log File****4.1.14.1.1 Purpose**

See 3.2.1

**4.1.14.1.2 Classifier AC-01****4.1.14.1.3 L.O.S.**

None

**4.1.14.2 Object Relation file****4.1.14.2.1 Purpose**

See 3.2.2

**4.1.14.2.2 Classifier AC-02**

**4.1.14.2.3 L.O.S.**

None

**4.1.14.3 Object name map**

**4.1.14.3.1 Purpose**

See 3.2.3

**4.1.14.3.2 Classifier AC-03**

**4.1.14.3.3 L.O.S.**

None

**4.1.14.4 Object tree file**

**4.1.14.4.1 Purpose**

See 3.2.4

**4.1.14.4.2 Classifier AC-04**

**4.1.14.4.3 L.O.S.**

None

**4.2 Behavior**

### 4.2.1 Generate object relation realization

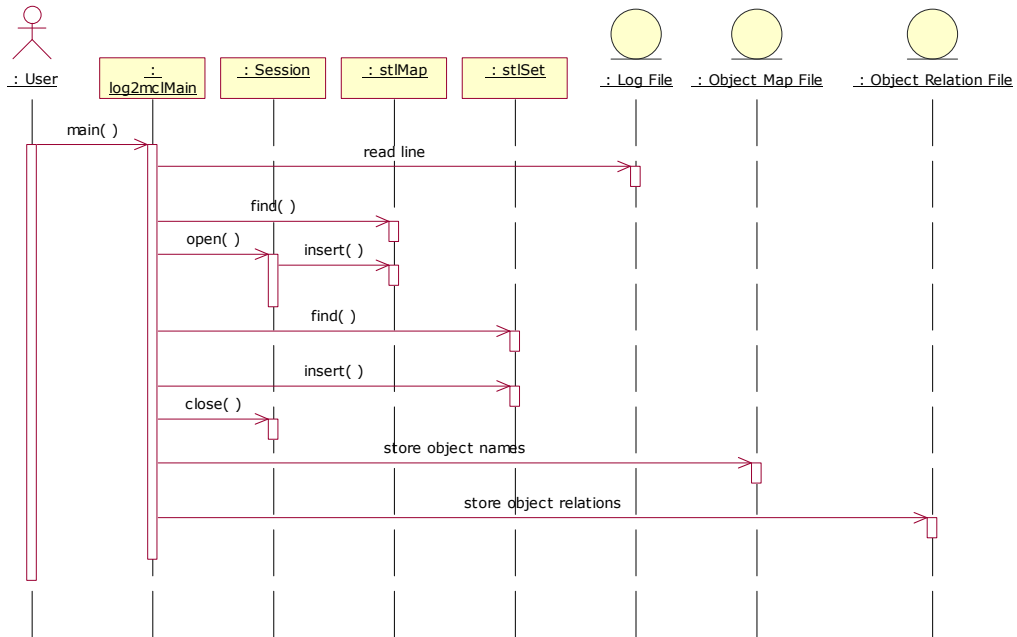


Figure 39 Generate object relation realization

### 4.2.2 Generate object tree realization

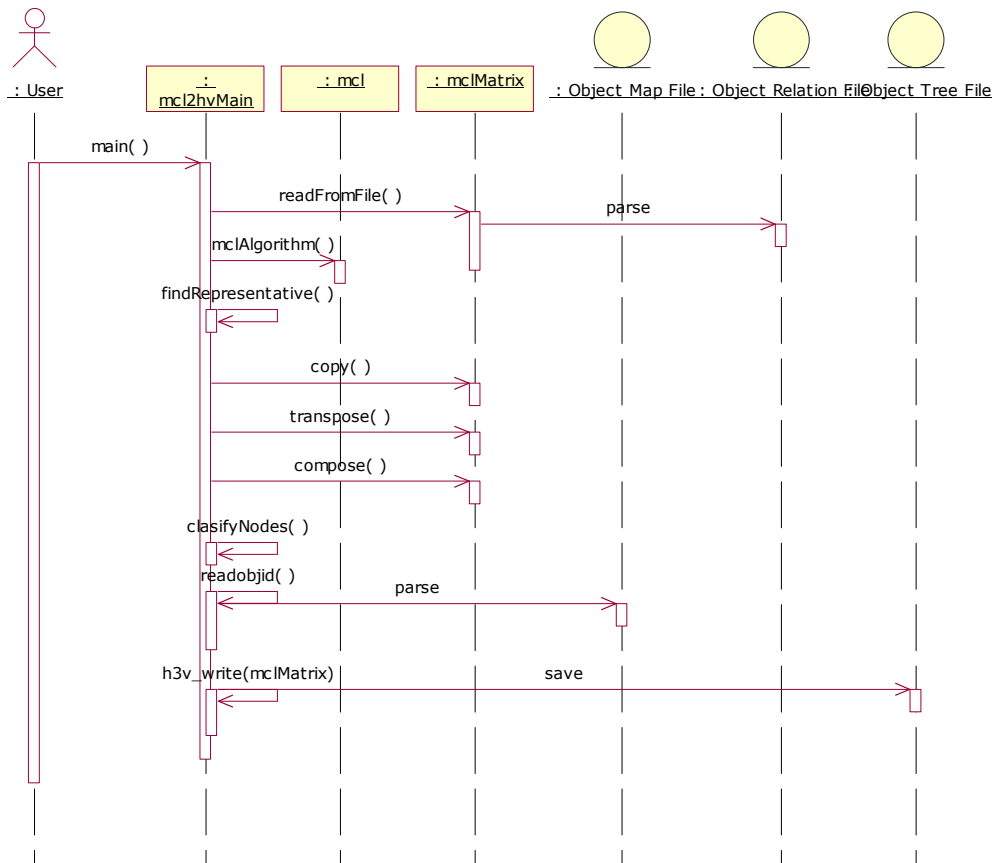
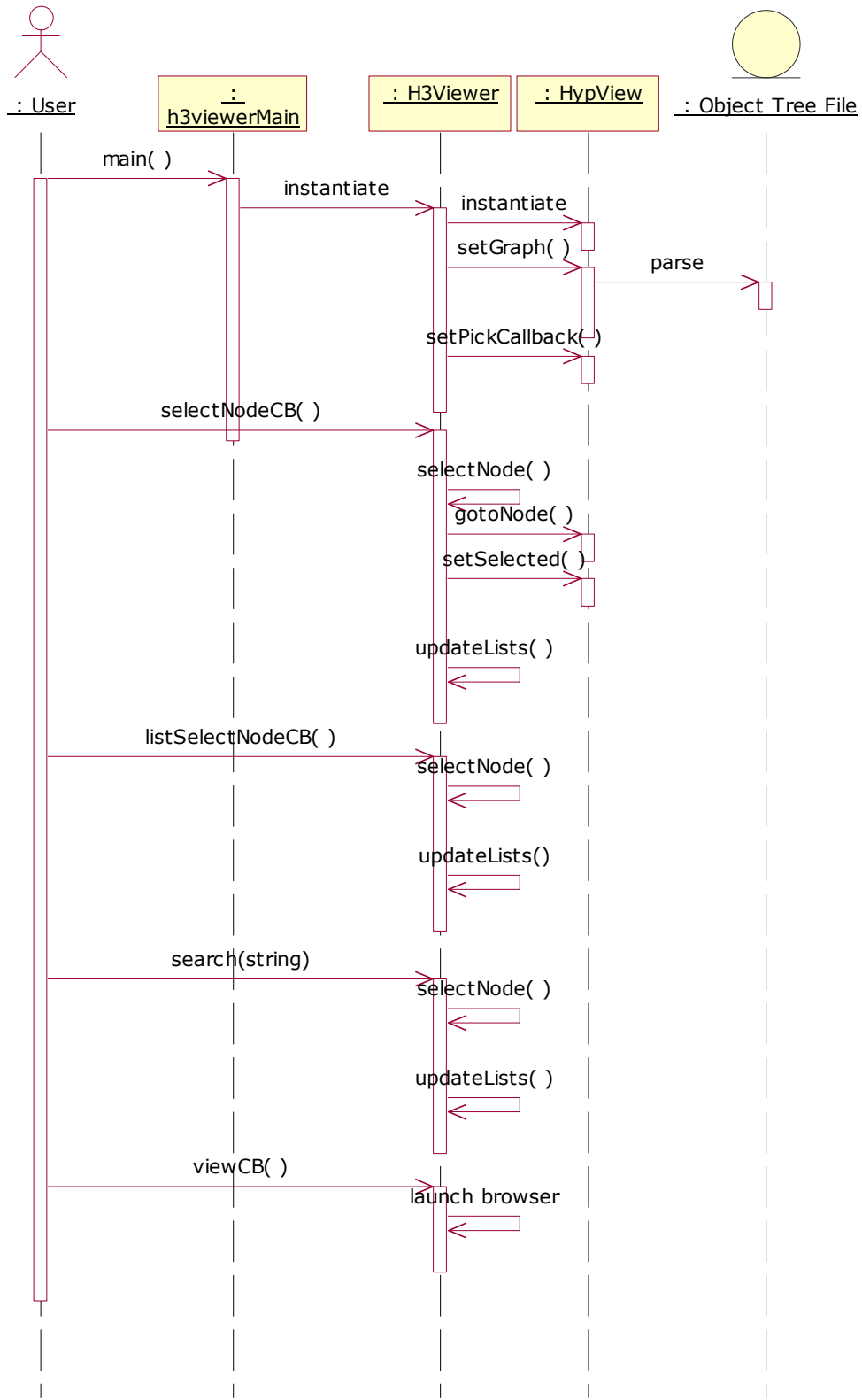


Figure 40 Generate object tree realization

### 4.2.3 Browse object tree realization



**Figure 41 Browse object tree realization**

### 4.3 L.O.S.

<b>L.O.S Goal</b>	<b>Applies To</b>	<b>How</b>	<b>Projected Value</b>	<b>Evaluation Technique</b>
LR-1 [SSRD 5]: Dependability	SCO-01, SCO -02, SCO -03, SCO -04	Equally	Pass 95% of test cases	Estimation
LR-2 [SSRD 5]: Usability	SCO -04	Equally	Satisfied with the client	Estimation
LR-3 [SSRD 5]: Operability in multitasking environment	SCO -03	Equally	Pass 95% of test cases	Estimation
LR-4[SSRD 5]: Performance on data of current scale	SCO -04	Equally	Able to perform properly with 10% increase.	Estimation

### 4.4 Patterns & Frameworks

See 3.4

## 4.5 Project Artifacts

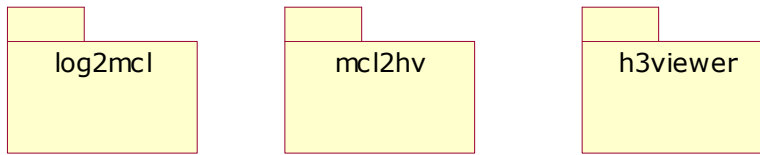


Figure 42 Project directory structure

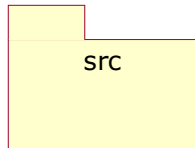


Figure 43 log2mcl directory structure

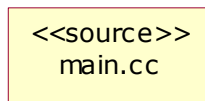


Figure 44 log2mcl/src modified files



Figure 45 mcl2hv directory structure

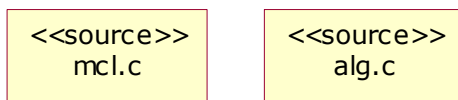


Figure 46 mcl2hv/src modified contents

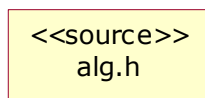
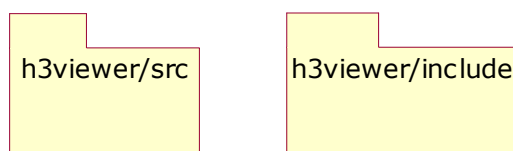


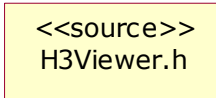
Figure 47 mcl2hv/include modified contents



**Figure 48 h3viewer directory structure**



**Figure 49 h3viewer/src modified contents**



**Figure 50 h3viewer/include modified contents**

## 5. Glossary for System Analysis and Design

**DA- Digital Archive :** Collection of electronic and multi-media documents, such as images, photos, audios, and videos

**GUI:** graphical user interface.

**Log Data:** they are the data gathered when users visit the library's digital achieves, they contain: user's IP address, item's id, and the time of visit.

**LANL :** Los Alamos National Laboratory.

**L.O.S –** Level of Service

**MBASE:** Model-based System Architecting and Software Engineering (MBASE).

**UML:** Unified Modeling Language (UML).

## **6. Appendices**

None