CEN 5011 – Advanced Software Engineering – Section U1

Modeling Environment for the Communication Virtual Machine

Final Document

Team 2
Lazaro Pi
Leandro Wong
Manasa Bharadwaj
Sandeep Varry
Jorge Jauregui
Marc Gauthier

Professor: Peter Clarke

12/04/2008

Abstract

Communication plays a critical role in every field of human endeavor. It is especially important in collaborating projects done by small teams. It is used in steps like brainstorming, sharing of findings, progress reports, and presentation of the results and thus, it can determine a research project's success or failure. The lack of fast and effective communication could greatly hinder a project's progress when team members are separated by large distances.

The purpose of Communication Modeling Environment (CME) is to allow members of those teams to create communication models that can be later executed on the Communication Virtual Machine (CVM). All of this while seamlessly integrating different communication systems to automate the process of switching between them to achieve cost efficiency or communication quality. To achieve this we analyzed the system in use at our client's facilities and developed a project plan for our proposed system. The CME will be created using model driven software development techniques in an easy to use graphical modeling framework, which will eventually allow us to deploy a communication application employing the features already present in applications like the Skype and Smack API's. In particular, we aim to support the creation of CML models in a graphical environment and the seamless conversion between GCML and XCML formats. We will be able to produce a complete schema instance in the CME. The system will be able to load and display schemas in the modeling environment. The system will also store models in GCML and XCML format, including layout information.

Table of figures

1.Introduction

This chapter constitutes an introduction to the Communication Modeling Environment (CME). In this chapter we will describe the system to be developed, namely a brief introduction of its purpose, the scope that it comprises, functional and nonfunctional requirements. It also presents a list of definitions, acronyms, abbreviations that will be used throughout the text and an overview of this document.

1.1. Purpose of System

The system is divided in two parts: the first one is a set of metamodels, tools, and graphic interfaces to describe and create communication models in GCML (for definitions of this and other words, please refer to section 1.3). The second part of the system comprises a set of transformations to convert between GCML and XCML formats.

The application can be run from the Eclipse Platform, and offers modelers the ability to create graphical communication models (GCML) that will be executed later. Modelers can drag shapes and edges to a canvas to describe those communication models, and save them for later use. The system will also be able to produce a complete schema instance in the modeling environment. The second purpose of the system is to transform models in GCML to schemas in XCML and vice versa.

With this system the user will also store models, in both formats, XCML and GCML. Moreover, the system will be able to load and display schemas in the modeling environment. In the load operation, layout information will be included. Some schema validation will be provided in this project.

For functionalities corresponding to:

1. Creating the model.
2. Loading and storing the model.
3. Transforming and validating the model
4. Security of System.

1.2. Scope of System

The system will be implemented in Java and will have two decoupled layers: Modeling Environment (ME) and Users Communication Interface (UCI).

This system deals with:

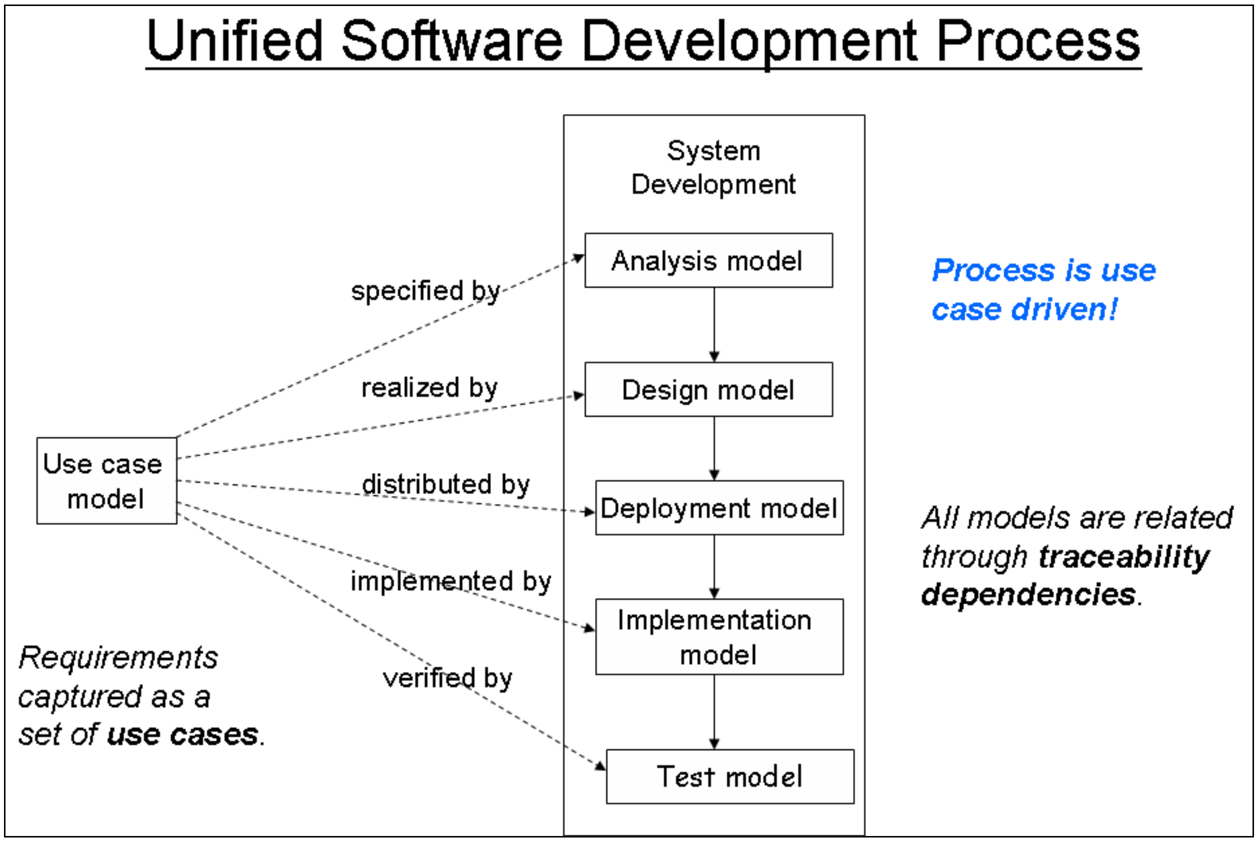
- The communication model creation in the graphical modeling environment.
- The conversion between XCML and GCML and vice versa.
- The storing and loading of previously saved models.
- Some facilities to provide security and administrative tools for the CME system and repository.

This system does not deal with:

- The basic software for model creation, the Eclipse Platform, which has to be installed separately
- The creation of accounts for the different communication systems. This has to be done by the user beforehand.
- The execution of models, this is done with the CVM.

1.3. Development Methodologies

The Unified Software Development Process model was chosen as the software process to follow while developing the RRComSSys (see Figure 1). This process allows for the encapsulation of both the functional and nonfunctional requirements into use cases. It provides a blueprint of the different stages of the development process and it forces traceability between them. The Software Requirement and Analysis Document (SRAD) culminated the Analysis model and provided the direction and foundation needed to develop the Design model. The architectural and design patterns were chosen after careful analysis of the use cases (as well as the diagrams) provided by the SRAD. The class and sequence diagrams in the Design model are an update of those provided by the SRAD. USDP allows the developers to follow a model driven approach. Each model in the USDP provides the foundation and outline for the following model. In the validation/verification the latest model provides the validation of the previous model. This evolutionary approach (rather than a revolutionary one) is what makes every model in the USDP so important and effective.



Since RRComSSys is a domain specific problem another development technique was incorporated. The Model-Driven Software Development (MDSD) was chosen to specify the domain of RRComSSys (see Figure 2). Following the approach of MDSD we defined the Metamodel (see section 4.2) to be used by the EMF & GMF framework. The transformations inside the User Communication Interface (see Chapter 4) from the creation of the G-CML to the calls using the SkypeAPI display the facilities that MDSD provides for the developer.

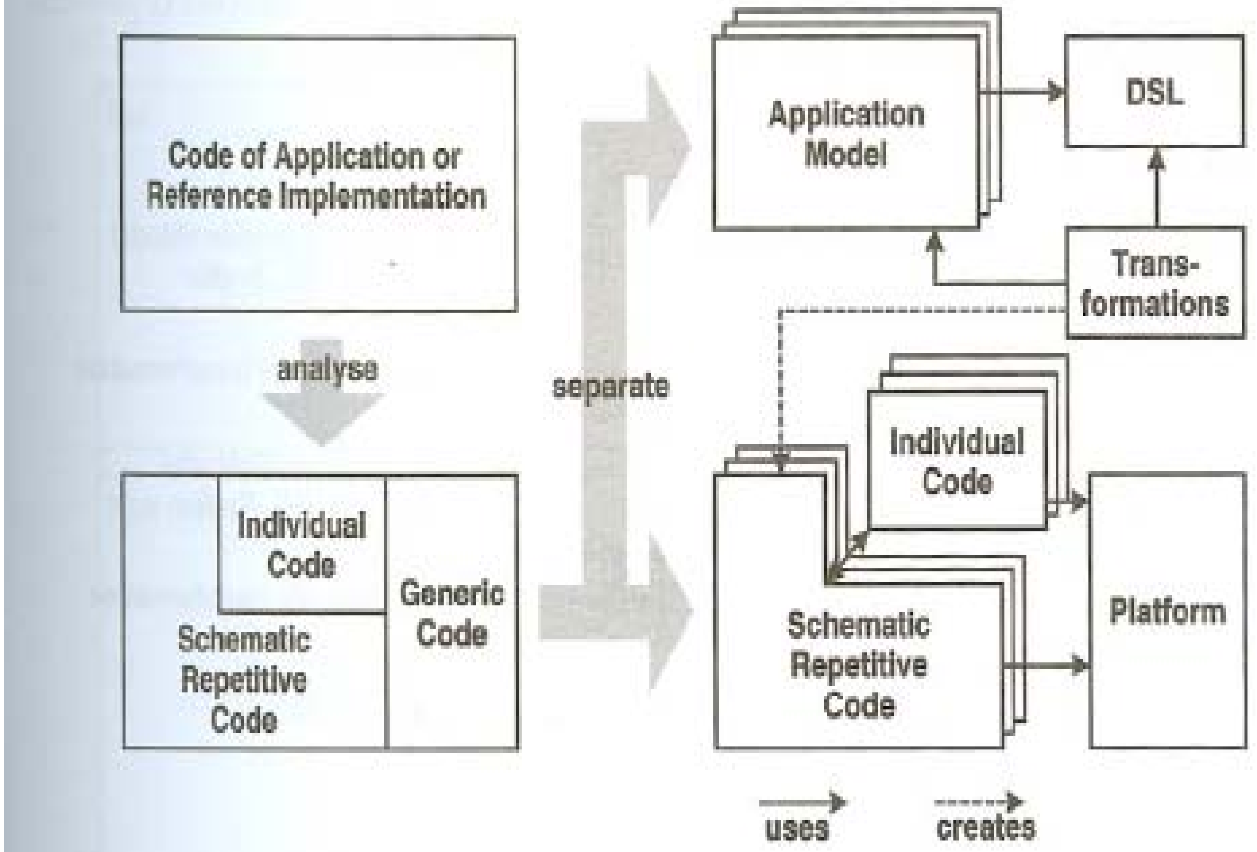
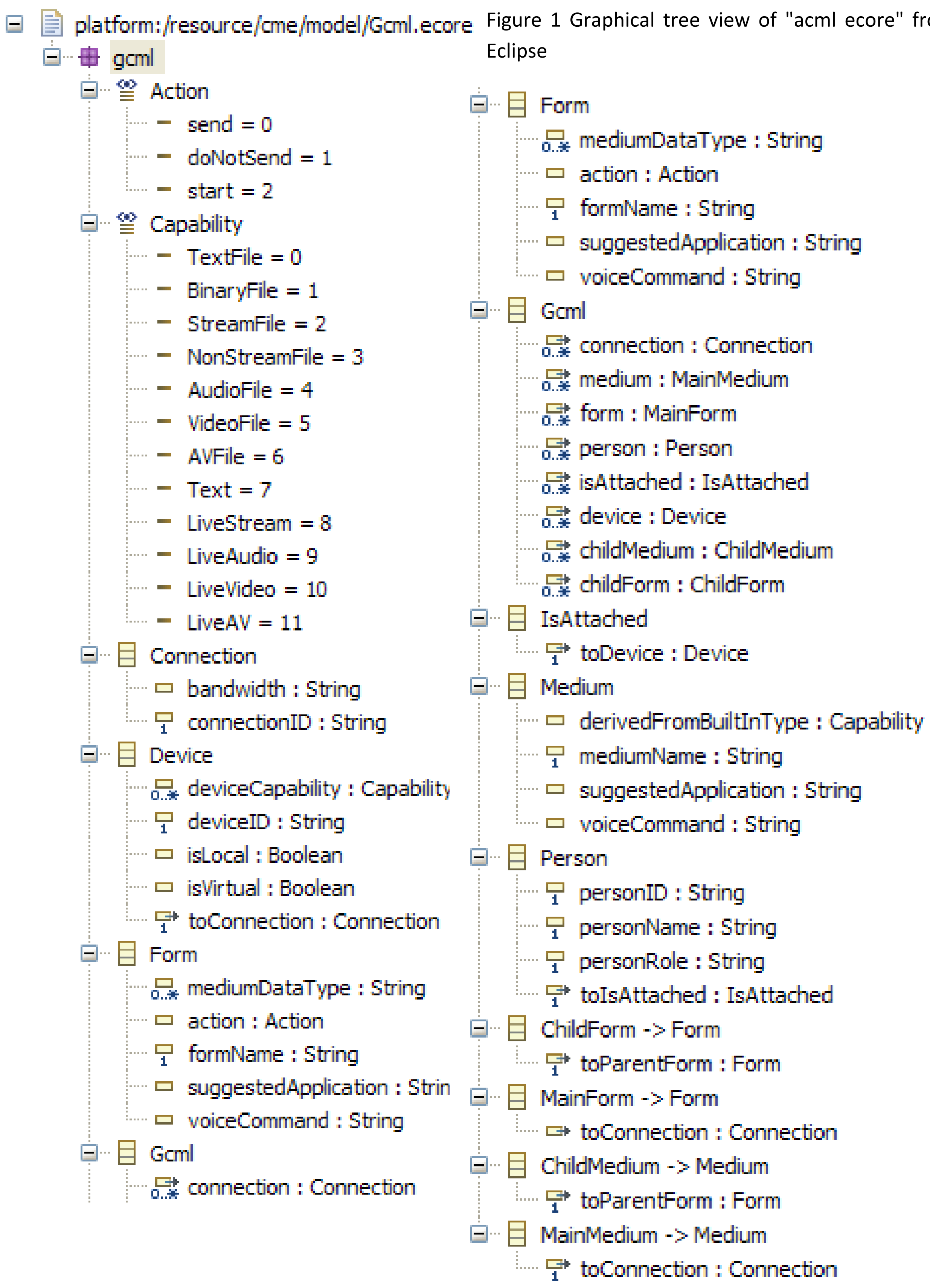


Figure 2.1 The basic ideas behind Model-Driven Software Development.

Gcml Metamodel

Figure 1 Graphical tree view of "acml ecore" from Eclipse



```
platform:/resource/cme/model/Gcml.ecore
  gcml
    Action
      send = 0
      doNotSend = 1
      start = 2
    Capability
      TextFile = 0
      BinaryFile = 1
      StreamFile = 2
      NonStreamFile = 3
      AudioFile = 4
      VideoFile = 5
      AVFile = 6
      Text = 7
      LiveStream = 8
      LiveAudio = 9
      LiveVideo = 10
      LiveAV = 11
    Connection
      bandwidth : String
      connectionID : String
    Device
      deviceCapability : Capability
      deviceID : String
      isLocal : Boolean
      isVirtual : Boolean
      toConnection : Connection
    Form
      mediumDataType : String
      action : Action
      formName : String
      suggestedApplication : Strin
      voiceCommand : String
    Gcml
      connection : Connection
```

```
    Form
      mediumDataType : String
      action : Action
      formName : String
      suggestedApplication : String
      voiceCommand : String
    Gcml
      connection : Connection
      medium : MainMedium
      form : MainForm
      person : Person
      isAttached : IsAttached
      device : Device
      childMedium : ChildMedium
      childForm : ChildForm
    IsAttached
      toDevice : Device
    Medium
      derivedFromBuiltInType : Capability
      mediumName : String
      suggestedApplication : String
      voiceCommand : String
    Person
      personID : String
      personName : String
      personRole : String
      toIsAttached : IsAttached
    ChildForm -> Form
      toParentForm : Form
    MainForm -> Form
      toConnection : Connection
    ChildMedium -> Medium
      toParentForm : Form
    MainMedium -> Medium
      toConnection : Connection
```

1.4. Definitions, Acronyms, and Abbreviations

CME: Communication Modeling Environment. The system described in this document. The CME allows a modeler to create graphical models of communication which can then be transformed into XCML.

Metamodel: A model that describes the possible structure of models – in an abstract way, it defines the constructs of a modeling language and their relationships, as well as the constraints and rules – but not the concrete syntax of the language.

GUI: The user interface allows people to interact with the computer which have graphical icons, visual indicators or special graphical elements representing the models and functionalities of the Modeling environment for CVM.

EMF: Eclipse Modeling Framework is an Eclipse-based modeling framework and code generation facility for building tools and other applications based on a structured data model.

UML: UML (Unified Modeling Language) is a language used to visualize, specify, construct, and document the artifacts of a software-intensive system.

Abstract Syntax: Specifies what the language structure looks like.

CML: Communication Modeling Language, a DSL to specify the Communication domain.

Concrete Syntax: Specifies what the parser of the language accepts.

CVM: Communication Virtual Machine

Domain: Bounded field of interest or knowledge

DSL: Domain Specific Language.

Formal Model: It is a sentence formulated in the DSL

GCML: Graphical CML. Communication Modeling Language represented as Graphs.

GEF: Graphical Editing Framework

GMF: Graphical Modeling Framework

MDA: Model Driven Architecture

MDSD: Model Driven Software Development

Meta Meta-Model: It is the meta-model of the meta-model

Meta-Model: Encompasses the abstract syntax and static semantic of a language.

NCB: Network Communication Broker

RCP: Eclipse Rich Client Platform. While the Eclipse platform is designed to serve as an open tools platform, it is architected so that its components could be used to build just about any client application. The minimal set of plug-ins needed to build a rich client application is collectively known as the **Rich Client Platform**.
Taken from http://wiki.eclipse.org/RCP_FAQ#What_is_the_Eclipse_Rich_Client_Platform.3F .

SE: Synthesis Engine

Self-Configuration: automatically configuring existing system components and integrating new components

Self-Optimization: automatically tuning resources

SRAD: Software Requirement and Analysis Document

Static Semantics: Determines the criteria for well-formation of a language

UCI: User Communication Interface.

USDP: Unified Software Development Process

CME: Communication Modeling Environment

XML: Extensible Markup Language

XCML: Communication Modeling Language represented in XML.

1.5. Overview of Document

Chapter 2 of the document, Current System, describes the existing system that provides a similar functionality for CML modeling. This system doesn't offer a transparent method of conversion between GCML and XCML formats.

Chapter 3 of the document, Project Plan, identifies the milestones of the projects and breaks it down in a number of subtasks. It also describes the hardware and software requirements and enumerates the different roles of the team members.

Chapter 4, Proposed System, is centered on the functionality of the developed system. It gives an overview, enumerates the functional and non-functional requirements and offers various models describing the static and dynamic views of the system, among them, Scenarios, Use Cases diagrams and Requirements analysis.

Chapter 5 consists Software Architecture and an overview of different subsystems like ME, Modeling Transformation and Validation, Repository and Package diagram. The detailed subsystem decomposition, also the software and hardware mapping and persistent data management is presented here.

Chapter 6 introduces object Interactions which is the sequence diagrams , control objects that is state-machines and detailed class design with class description.
Chapter 7 introduces with testing and presents a list of system test case, sub system tests and unit tests. Chapter 8 consists of a glossary of terms, and Chapter 6 provides a Software development agreement to be signed by the interested parts before the system goes into development.

2. Current System

Currently there are many services that provide communication capabilities to users. But, the majority does not offer automated sequence of services. If a developer attends to his needs, he can implement this functionality. Consequently, if this user now needs something else, it becomes very inefficient to constantly change the implementation.

The CVM accepts XCML models that can be developed manually by a knowledgeable modeler. There are several prototypes of graphical modeling environments to obtain GCML versions of models but they lack some of the functionalities required by our clients. A complete Modeling Environment is unavailable at this time. There is no system capable of loading, storing and displaying complete schemas. Currently there is not system capable of transforming models in G-CML to schemas in X-CML. Our system will incorporate a solution for all these deficiencies.

3.Project Plan

The following sections show the tasks required to organize and schedule the project's Requirements and Analysis, Design Development and Implementation, and Testing.
Section 3.1, Project Organization, describes the assignment of roles during the project. Section 3.2, Hardware and Software Requirements, provides a set of system requirements for both the software used to develop the system and the deployment system to house the software. Finally, Section 3.3, Work Breakdown, will identify a set of milestones and deliverables required for the completion of the project.

3.1. Project Organization

The Project Organization section defines the roles for each individual in the project.

Deliverable 1:

Team Leader: Lazaro Pi.

System Manager: Jorge Jauregui.

Implementers: Marc Gauthier, Sandeep Varry.
Project Manager: Leandro Wong.

Document Editor: Manasa Bharadwaj.

Time Keeper: Roberto Espinosa.

Note Taker: Roberto Espinosa.

Deliverable 2:

Team Leader: Manasa Bharadwaj.

System Manager: Lazaro Pi.

Implementers: Leandro Wong, Marc Gauthier.
Project Manager: Robert Espinosa.

Document Editor: Sandeep Varry.

Object Designer: Leandro Wong.

Time Keeper: Jorge Jauregui.

Note Taker: Marc Gauthier.

Deliverable 3:

Team Leader: Sandeep Varry.

System Manager: Leandro Wong.

Implementers: Jorge Jauregui, Lazaro Pi, Leandro Wong.
Project Manager: Manasa Bharadwaj.

Document Editor: Jorge Jauregui.

Object Designer: Marc Gauthier.

Time Keeper: Manasa Bharadwaj.

Note Taker: Manasa Bharadwaj.

3.2. Hardware and Software Requirements

To develop the CME we have decided to use the Eclipse Modeling Framework (EMF) v2.3. The following are the software requirements for EMF:

- JDK 5.0 or later
- Eclipse Europa v3.3.0 or later
- EMF v2.3 Plugin or later

The following are the system requirements for development:

- Computer with a 1.6 GHz or faster processor
- 384 MB of RAM or more (768 MB of RAM or more for Windows Vista)
- 2.2 GB of available hard-disk space
- 5400 RPM or faster hard drive
- 1024 x 768 or higher-resolution display

Any of the following operating systems:

- Windows Vista® (x86 & x64) - all editions except Starter Edition
- Windows® XP (x86 & x64) with Service Pack 2 or later - all editions except Starter Edition
- Windows Server® 2003 (x86 & x64) with Service Pack 1 or later (all editions)
- Windows Server 2003 R2 (x86 and x64) or later (all editions)

The following are system requirements to execute CME application:

- Computer with a 1.0 GHz or faster processor
- 256 MB of RAM or more
- 50 MB of available hard-disk space
- 800 x 600 or higher-resolution display
- JDK 5.0 or later

The following software packages are utilized while documenting/developing the project:

- StarUML
- Microsoft Word
- Microsoft Project
- Eclipse Europa v3.3.0

3.3. Work Breakdown

The following table shows each of the tasks related to this project and how they are scheduled.

Task	Milestone	Task Name	Start	Finish	Predec.	Duration
1	No	Analysis/Software Requirements	9/3/2008	10/6/2008		24 days
2	No	Generate Use Cases	9/9/2008	9/15/2008		5 days
3	No	Primary Project Schedule	9/3/2008	9/8/2008		4 days
4	No	Identify Hardware, Software Resources	9/16/2008	9/22/2008	2	5 days
5	Yes	Review Use Cases	9/16/2008	9/16/2008	2	1 day
6	No	Identify milestones	9/16/2008	9/17/2008	2,5	1.5 days
7	No	Create Cost Analysis	9/17/2008	9/22/2008	5	4 days
8	No	Obtain approvals to proceed	9/22/2008	9/22/2008	7	1 day
9	No	Create Object and Dynamic diagrams	9/16/2008	9/18/2008	5	2.5 days
10	Yes	Create SRD deliverable	10/1/2008	10/2/2008	5	1.5 days
11	No	Create Presentation	10/6/2008	10/6/2008	10	1 day
12	No	Design	10/7/2008	11/3/2008		20 days
13	No	Review software specifications	10/7/2008	10/10/2008	11	3.5 days
14	No	Create Architectural Design	10/9/2008	10/15/2008	13	4 days
15	No	Create Object Design	10/9/2008	10/23/2008	13	10 days
16	Yes	Review Design with Team	10/23/2008	10/28/2008	15,14	3 days
17	No	Obtain approval to proceed	10/28/2008	10/28/2008	16	0.5 days
18	Yes	Create DD deliverable	11/3/2008	11/3/2008	17	0 days
19	No	Implementation	10/28/2008	11/27/2008		22.5 days
20	No	Review System / Object Design	10/28/2008	10/31/2008	16	3.5 days
21	Yes	Implement Import module	10/28/2008	11/25/2008	16	20 days
22	Yes	Implement Convert	10/28/2008	11/11/2008	16	10 days
23	No	Unit Testing	10/31/2008	11/6/2008	16FS+3	4 days
24	No	Functional Testing	11/4/2008	11/27/2008	21FS-75%	17 days
25	No	Development complete	11/27/2008	11/27/2008	24	0.5 days
26	No	Final Deliverable	11/4/2008	12/1/2008		20 days
27	No	Develop Final Software documentation	11/4/2008	11/10/2008	12	5 days
28	No	Review of Final Deliverable	11/11/2008	11/11/2008	27	0.5 days
29	No	Final Presentation	11/28/2008	12/1/2008	19	2 days

3.4. Cost Estimate

The estimate is based on COCOMO II. Work is breakdown based on the different aspects of the software process. COCOMO II allows the user to estimate the cost based on functions points. COCOMO II then converts the function points into “Lines of Codes” and Person Months. A Complete estimate with a complete analysis of each phase is provided in Appendix F.

```
Project Name:   CVM

Scale Factors:      PREC    FLEX    RESL    TEAM    PMAT
                       L       N       N       N       N
                    4.96    3.04    4.24    3.29    4.68


                                       NOM     ACT              RATE
                 Module              Effort  Effort   PROD         &   INST
  Module Name     Size    EAF         DEV     DEV               COST   COST    Staff
RISK

                       FP                                      100.0
            SRD     23539   1.00     117.1     117.1 201.1     11707    0.5      4.2
0.0

                       FP                                      100.0
             DD      6958   1.00      34.6      34.6 201.1      3461    0.5      1.3
0.0

                       FP                                      100.0
             IM     42294   1.00     210.4     210.4 201.1     21035    0.5      7.6
0.0

                       FP                                      100.0
             RE     36040   1.00     179.2     179.2 201.1     17925    0.5      6.5
0.0

    TOTAL SLOC   108831     OPTIMISTIC   362.7 301.6     36266    0.3    15.0
0.0
                            MOST LIKELY  541.3 201.1     54128    0.5    19.6
0.0
                            PESSIMISTIC  811.9 134.7     81192    0.7    25.9
0.0

                 OPTIMISTIC    MOST LIKELY   PESSIMISTIC
  SCHEDULE             24.3           27.6          31.4
```

4. Requirements of the System

This chapter first lists the functional and nonfunctional requirements of the system which are detailed through the use of structured natural language narratives in the form of Use Cases. It then presents some scenarios, object and sequence diagrams illustrating the key functionalities of the CME system.

4.1. Functional and Non-Functional Requirements

4.1.1. For functionalities corresponding to creating the model

A) The system shall allow model creators to drag and drop objects to canvas and save model to repository. (See use case DragObjectCanvas Appendix B)

Constraints:

- The system shall have at most 1 drag or drop failure per 100 drags or drops of objects.
- The created model shall be saved within 2 seconds.

B.)The system shall allow model creators to create edge between objects and save model to repository. (See use case Create Edge between Objects in Appendix B)

Constraints:

- The system shall have at most 1 create edge failure per 100 creations of objects.
- The created model shall be saved within 2 seconds.

C) The system shall allow model creators to create Group Chat Model, TwoWay Voice Communication Model and File Transfer Model and set attributes to the model. (See use case Group Chat (Create), TwoWayVoice, FileTransfer in Appendix B).

Constraints:

- On the user interfaces, all the shapes must contain brief textual descriptions.
- The created model shall be saved within 2 seconds.

D) The systems shall allow model creators to create new Empty Model and Generic Model (See use cases CreateEmptyModel and CreateGenericModel in Appendix B)

Constraints:

The created model shall be saved within 2 seconds.

If Modeler enters a name that already exists. The CME should warn the modeler that this action will overwrite the exiting model.

E) The systems shall allow modeler to open existing GCML model (See use case OpenExistingModel in Appendix B)
Constraints:
The model shall be displayed within 10seconds.

4.1.2. For functionalities corresponding to loading and storing the model

F) The system shall allow users to create a Model and store it in the repository which can be used later. (See use case AddModelToRepository in Appendix B)
Constraints:
The Modeler will add the existing model to the repository structure and will save additional metadata to describe the model.
The Modeler can enter a name that already exists. System should overwrite the existing model with the same name.
G) The system shall allow users to import an XCML or GCML file into the CME by picking the file on the File System (See use case ImportModelFromXCML and ImportModelFromXCML in Appendix B)
Constraints:
If the file name to be imported already exists (GCML, XCML or both), the user will be prompted to enter a different file name.

H) The system shall allow user to create a duplicate copy of the repository. (See use case CreateMirrorBackupOfRepository in Appendix B)
Constrains:
Mean Time to Failure no more than: – 1 failure for every 160 hours of operation.
Should always be available to the administrator: -Down time of the CME fewer than 30 minutes every 80 hours of operation.

I) The system shall allow Sensitive attribute values to be encrypted in the repository file
(See use case EncryptSensitveData in Appendix B)

Constraints:

CME will take less than .005 seconds to encrypt sensitive data.

Should be always available to the administrator. Down time of the CME fewer than 30 minutes every 80 hours of operation

J) The system shall allow transformation of model from GCML to XCML (see use case ConvertFromGCMLtoXCML, TransformObjectData and GenerateLayoutData in Appendix B)

Constraints

4.1.3. For functionalities corresponding to Transforming and validating the model

K) The system shall allow transformation of model from GCML to XCML (see use case ConvertFromGCMLtoXCML in Appendix B)

Constraints:

Must take less than 15 seconds for transformations.

System should notify the user of a failure (incorrect format) gracefully without aborting the program.

L) The system shall transformation of model from XCML to GCML (see use case ConvertFromXCMLtoGCML, in Appendix B)

Constraints:

Transformations must take less than 15 seconds

System should notify the user of a failure (incorrect format) gracefully without aborting the program.

M) The System Shall , Transform Object Data and Generate Layout Data(See use cases , TransformObjectData and GenerateLayoutData in Appendix B)

Constrains:

Must take less than 0.45 second for object data

Mean Time to Failure no more than 1 failure for every 120 hours of operation is acceptable

System should notify the user of a failure (incorrect format) gracefully without aborting the program

N) The system shall Calculate Shape and Size, Calculate Coordinates of objects(see use cases CalculateShapeandSize, Calculate Coordinates in Appendix B)
Constraints:
Should take less than .25 second for calculations of size and shape.
Must take less than .50 second for calculation of coordinates.
System should notify the user of a failure (incorrect format) gracefully without aborting the program

O) The system shall Check Overlap of objects (See use case CheckOverlap in Appendix B)
Constraints:
Should take less than .25 second to check overlap.
System should notify the user of a failure (incorrect format) gracefully without aborting the program

P) The system shall allow validation of model. (See use cases ValidateModel in Appendix B)
Constraints:
Must take less than .50 seconds for validation.
System should notify the user of a failure (incorrect format) gracefully without aborting the program

Q) The system shall Check model Schema and Check Semantic Rules(see use case CheckModelSchema, and CheckSemanticRules in Appendix B)
Constraints:
Must take less than .50 seconds for validation.
System should notify the user of a failure (incorrect format) gracefully without aborting the program

4.1.4. For functionalities corresponding to Security of System.

R) The system shall allow Users s securely log into and log out the system. (See use cases LogIn and LogOut in Appendix B)

Constraints:

On the user interfaces, all controls that require input from the user must contain brief textual descriptions of their purpose.

The login and logout process shall be completed within 2 seconds.

S) The system shall allow administrator to create account, change password, delete user account and suspend user account (see use case in Appendix B)

Constraints:

User must be able to access the system within 3 seconds of pressing the Add User button or Change password button.

User account should be deleted within 2 seconds of pressing delete user button.

The account should be suspended within 2 seconds of pressing suspend user button.

T) The system shall lock and unlock running application (see use case LockRunningApplication and UnlockApplication in Appendix B)

Constraints:

User must be able to exit the system within 2 seconds of pressing the OK button

Mean Time to Failure – 1 failure for every 160 hours of operation is acceptable.

A. The system shall allow Users s securely log into and log out the system. (See use cases Log In and Log Out in Appendix B) Constraints:

• On the user interfaces, all controls that require input from the user must contain brief textual descriptions of their purpose.

• A help facility must also be provided. Furthermore, at least 75% of the testing panel of users must agree that the help facility has the same visibility as other controls on the form.

• The login and logout process shall be completed within 2 seconds.

4.2. Use Case Diagrams

High level Use case Diagram (packages)

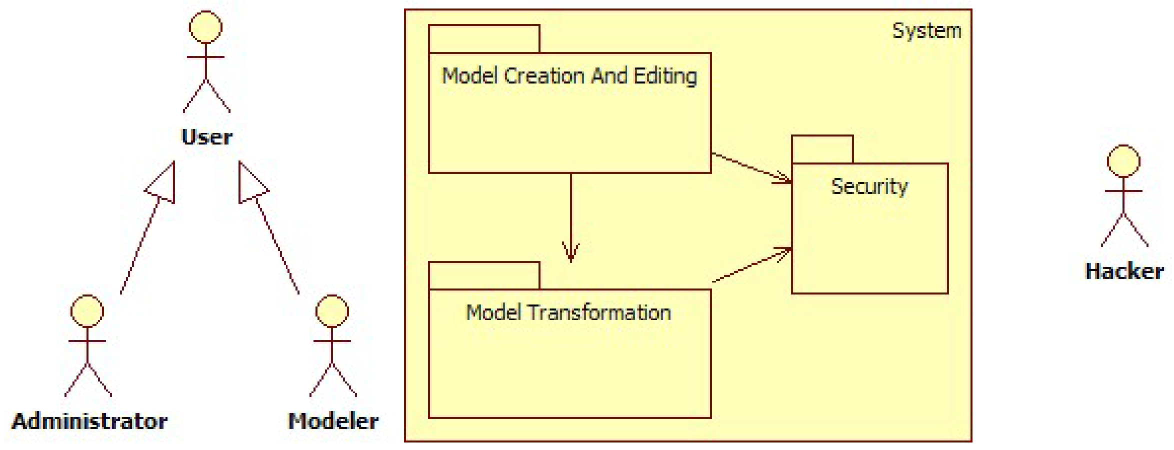


Figure 2 High level use case diagram

Model Creation and Editing

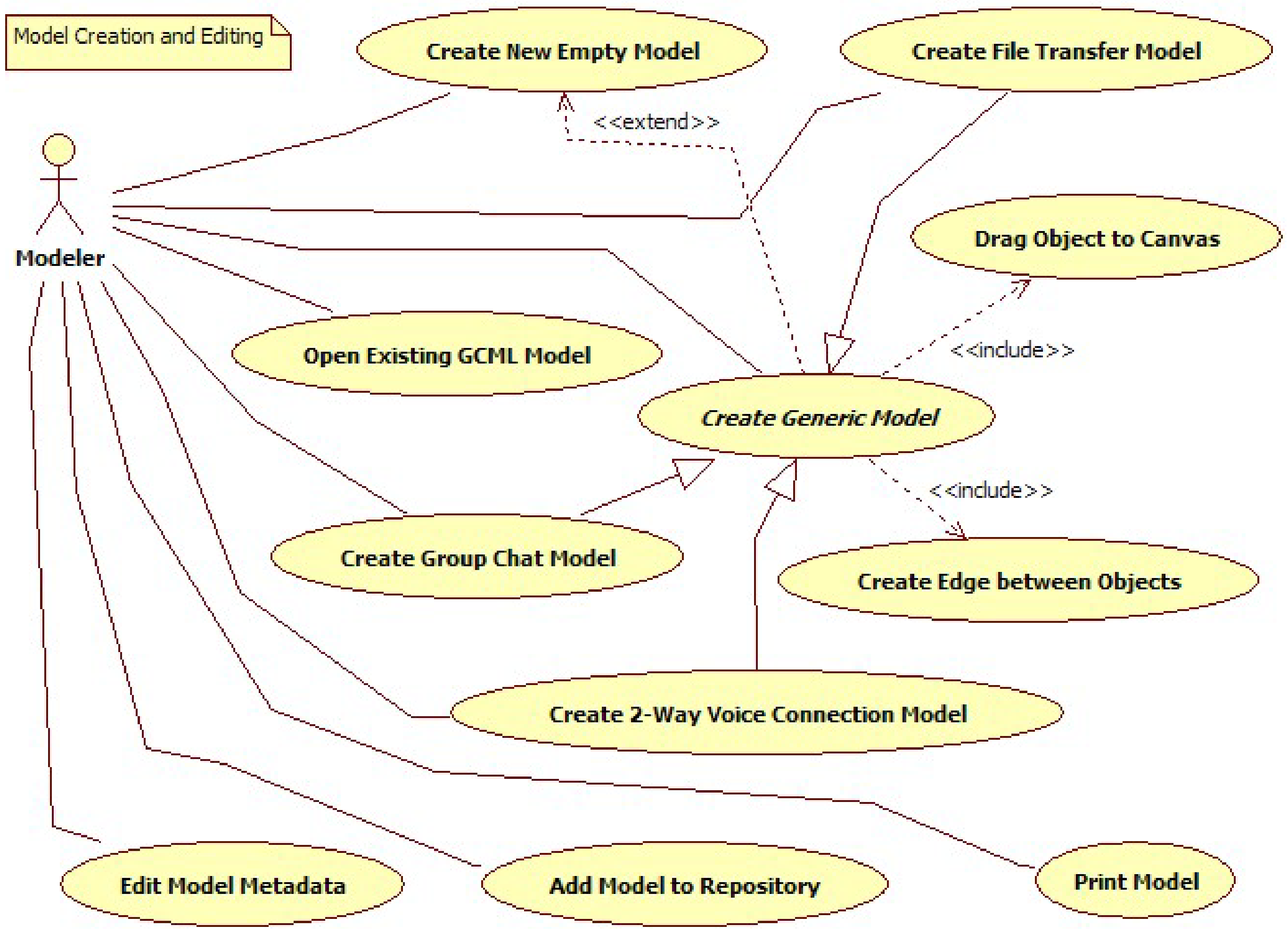


Figure 3 Model creation and Editing

Model Transformation

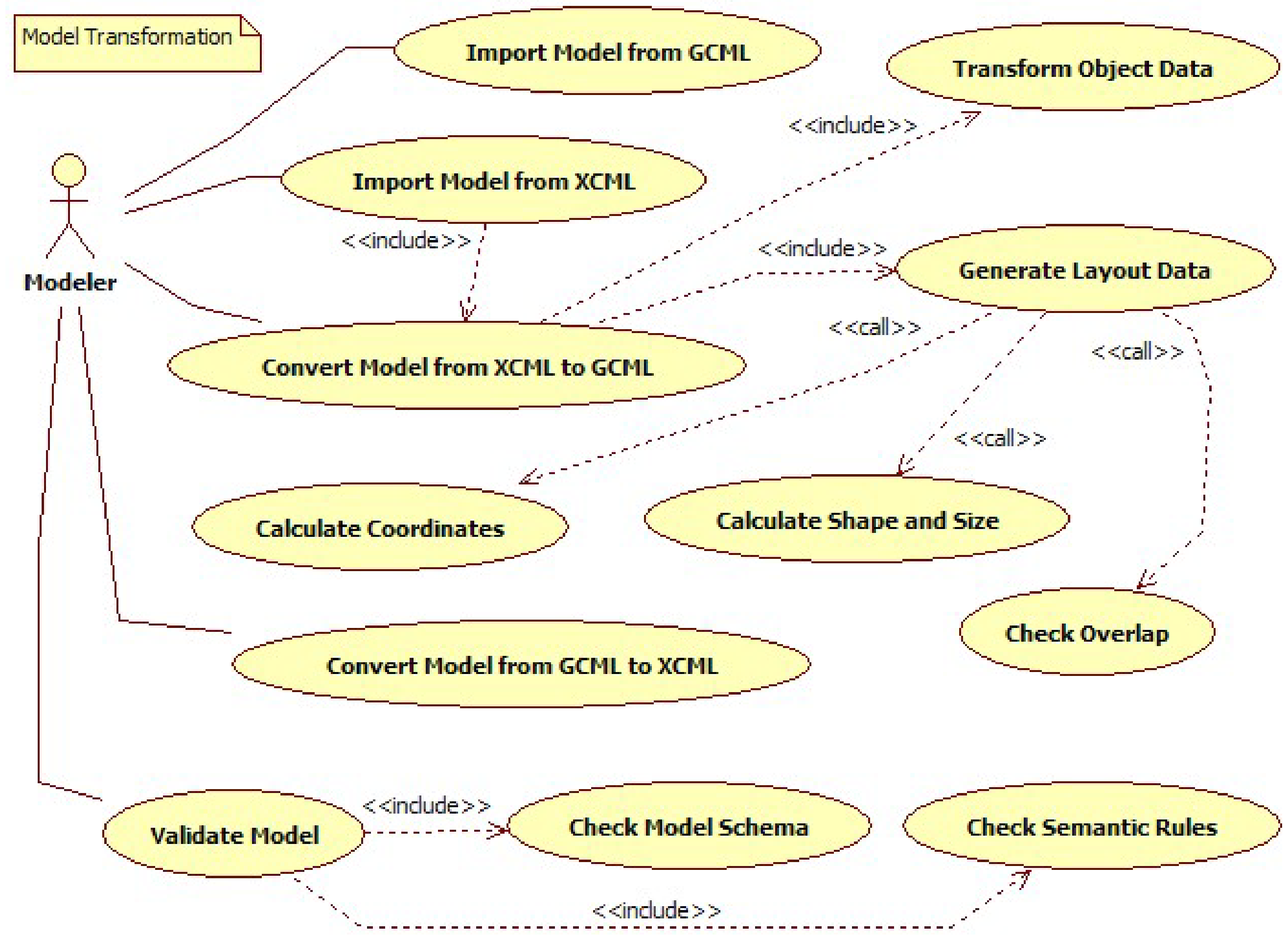
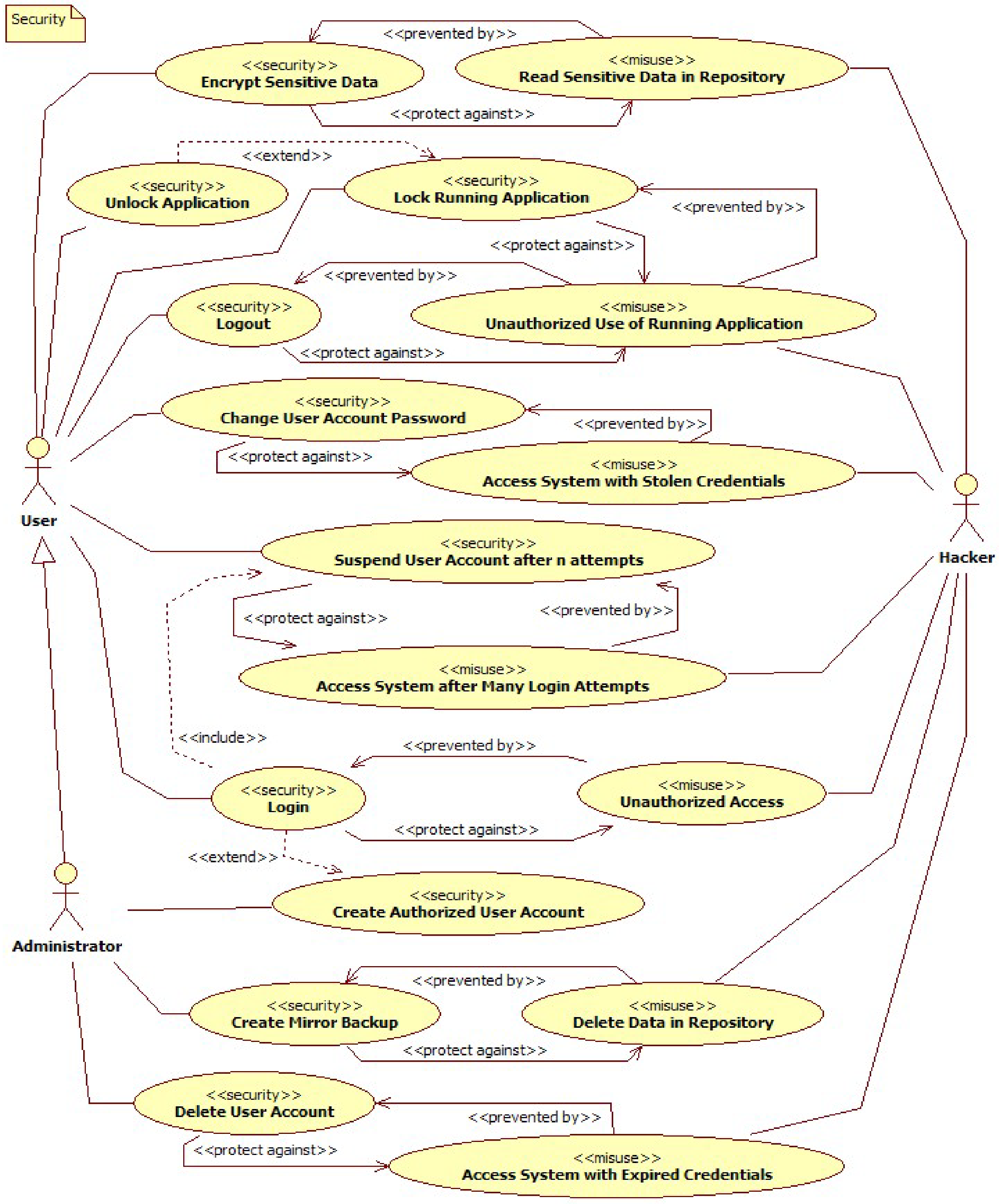


Figure 4 - Model Transformation schema

Security



Security
<<security>> Encrypt Sensitive Data
<<misuse>> Read Sensitive Data in Repository
<<prevented by>>
<<protect against>>
<<extend>>
<<security>> Unlock Application
<<security>> Lock Running Application
<<prevented by>>
<<protect against>>
<<prevented by>>
<<security>> Logout
<<misuse>> Unauthorized Use of Running Application
<<protect against>>
<<security>> Change User Account Password
<<prevented by>>
<<protect against>>
<<misuse>> Access System with Stolen Credentials
User
Hacker
<<security>> Suspend User Account after n attempts
<<protect against>>
<<prevented by>>
<<misuse>> Access System after Many Login Attempts
<<include>>
<<prevented by>>
<<security>> Login
<<misuse>> Unauthorized Access
<<protect against>>
<<extend>>
<<security>> Create Authorized User Account
Administrator
<<prevented by>>
<<security>> Create Mirror Backup
<<misuse>> Delete Data in Repository
<<protect against>>
<<security>> Delete User Account
<<prevented by>>
<<protect against>>
<<misuse>> Access System with Expired Credentials

Figure 5 Security

Use cases for Implemented Scenarios

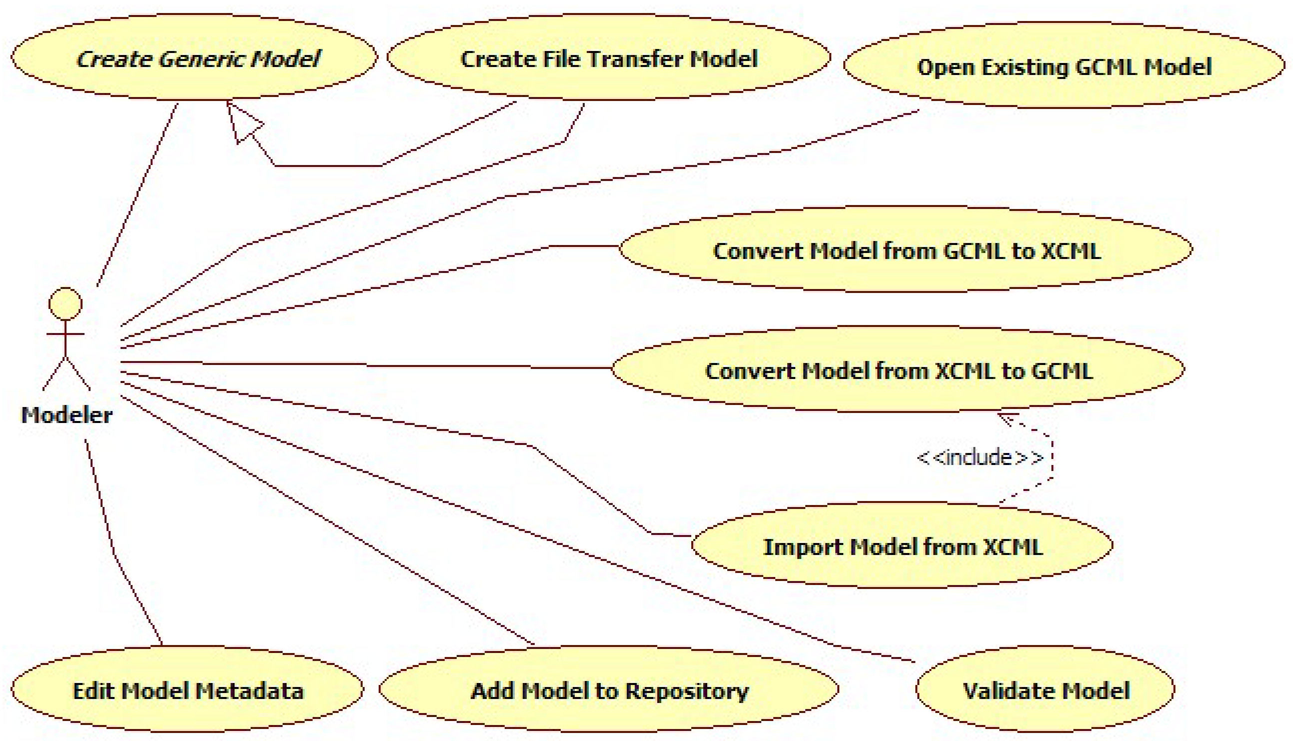


Figure 6 Use cases for implemented scenarios

4.2.1. Use Case Descriptions

Explanations:

Actors: Modeler and Administrator. For misuse cases: Hacker

Use Case ID: The use case ID has been coded as follows:

Team + "_" + SystemName +"_" + UseCaseLevel + "_" + IncrementalNumericID + "_" + UseCaseName

Example: T2_CME_SUB_01_CreateModel

This can be interpreted as a use case by Team 2 for the Communication Modeling Environment system. This is a functional sub-use case, has numeric id 01, and relates to the creation of a model.

Use Case Level Abbreviations:

HIGH = High Level Use Case

SYSE2E = System Level End to End Use Case

SUB = Functional Sub-use Case

4.3. Requirement Analysis.

The Requirements Analysis was partly performed by extracting requirements from Researchers involved in the CVM project. System details discussed in the CVM meetings were incorporated into the development of the UCM layer. Additional requirements were obtained from documents and publications that were produced outside of class by the group responsible for the CVM project. Examples of such documents are "CVM – A Communication Virtual Machine" by Deng et al, "UCM State Machine", and "UCM Control Script". Any assumptions were verified with the professor and the parties involved in the CVM project

Also we obtained the background of the CVM from the latest version of CVM provided to us by our clients CVM V1. Repeatedly the requirements were refined with our better understanding and constant communication with our clients.

5. Software Architecture

Architectural patterns are used to help describe how the system is decomposed, the tasks to be performed by the system's components, as well as the interrelations between them. This section details the proposed software architectures for the CME; which are Model-View-Controller (MVC) and Repository and Pipe and Filter as subordinate architectures. A package diagram showing the major subsystems, an overview of each subsystem, reasons for our choices of architecture, and metamodels for the DSL are among the subjects contained in this section. This section of the document also provides a UML profile that represents the architecture, details of the architecture to platform transformation and a detailed description of each major subsystem in the system.

5.1. Overview

The system architecture is based in the MVC architecture pattern with two sub patterns: the pipe and filter and the repository.

Modeling Environment: Model-View-Controller

The architecture of the system is MVC. This pattern accommodates perfectly to be run on top of the Eclipse Rich Client Platform (RCP), which is our choice implementation. CML is our representation of the communication domain and acts as our metamodels. There are two different models based on this metamodels: Xcml and Gcml. The "gcml" package inside the cme subsystem contains all the classes representing the Gcml model.

The Gcml model is directly manipulated through the modeling environment and therefore acts as the model for the MVC implemented in the Eclipse platform. The classes to accomplish this are located in the cme.diagram and cme.edit subsystems. Some of these classes represent the different part needed to create the graphical view of the models while others provide a set of controller objects to manipulate and edit the models (in several categories, such as commands, editors, parsers, etc.). These controller objects depend on the EMF, GEF, and GMF components and ultimately in the eclipse RCP to function. When we present a high level view of these controllers we will refer to it as the CmeController.

There are two views of the Gcml model: a graphical view (where the modeler can see shapes and arrows representing the different entities and they can edit properties of those entities) and a textual view (where the modeler can open the XML text directly. This textual view is saved in the repository for future use and is divided in two parts: object model data (files with .gcml extension) and layout data (files with .gcml_diagram extension). The Modeling Environment (cme.diagram and cme.edit running on top of the Eclipse RCP) depends on the Transformation subsystem and the Repository subsystem for some of its functionalities. A separate Administrative facility contains all code needed to support administrative features and security (called cme.admin).

Model Transformation and Validation: Pipe and Filter

The pipe and filter architecture is used in the UCI component to accomplish transformation and validation of models. The transformation package depends on the Gcml and the Xcml object models (in order to produce models in both representations). The transformations are organized as a series of filters or steps where the input of one step is the output of the previous one. The client code provides the initial input and obtains the final output from the pipeline. This standard way of implementing the transformation steps should allow us to change the order of the filters, add new filters, and introduce a pause in the execution by stopping in an intermediate step, when the requirements change. The client code is virtually unaffected. Validation is done in a similar way although the set of filters is different. The UCI package contains two Control classes as entry points to the services provided: Transform and Validate. These classes initialize the appropriate pipeline, executes the filters and returns the final result (if any is needed) back to the user.
Refer to Appendix B Use Case 10.2.1.9(Convert the model form GCML to XCML)
Use Case 10.2.1.10 (Convert model form XCML to GCML)
Use Case 10.2.1.19 (Validate Model)
Use Case 10.2.1.20 (Check Model Schema)
Use Case 10.2.1.21(Check Semantic Rule)

Repository

The system also uses a central repository for the models, the intermediate input and output results of the pipeline, and application configuration information. The repository package, containing a group of repository classes, controls the access to the repository and exposes an interface to the rest of the application which makes it easier to change the actual repository (database, text files, xml files, etc.) used with minimal effect on the rest of the application.
Refer to Appendix B Use Case 10.2.1.11 (Add Model to the Repository)

Package Diagram

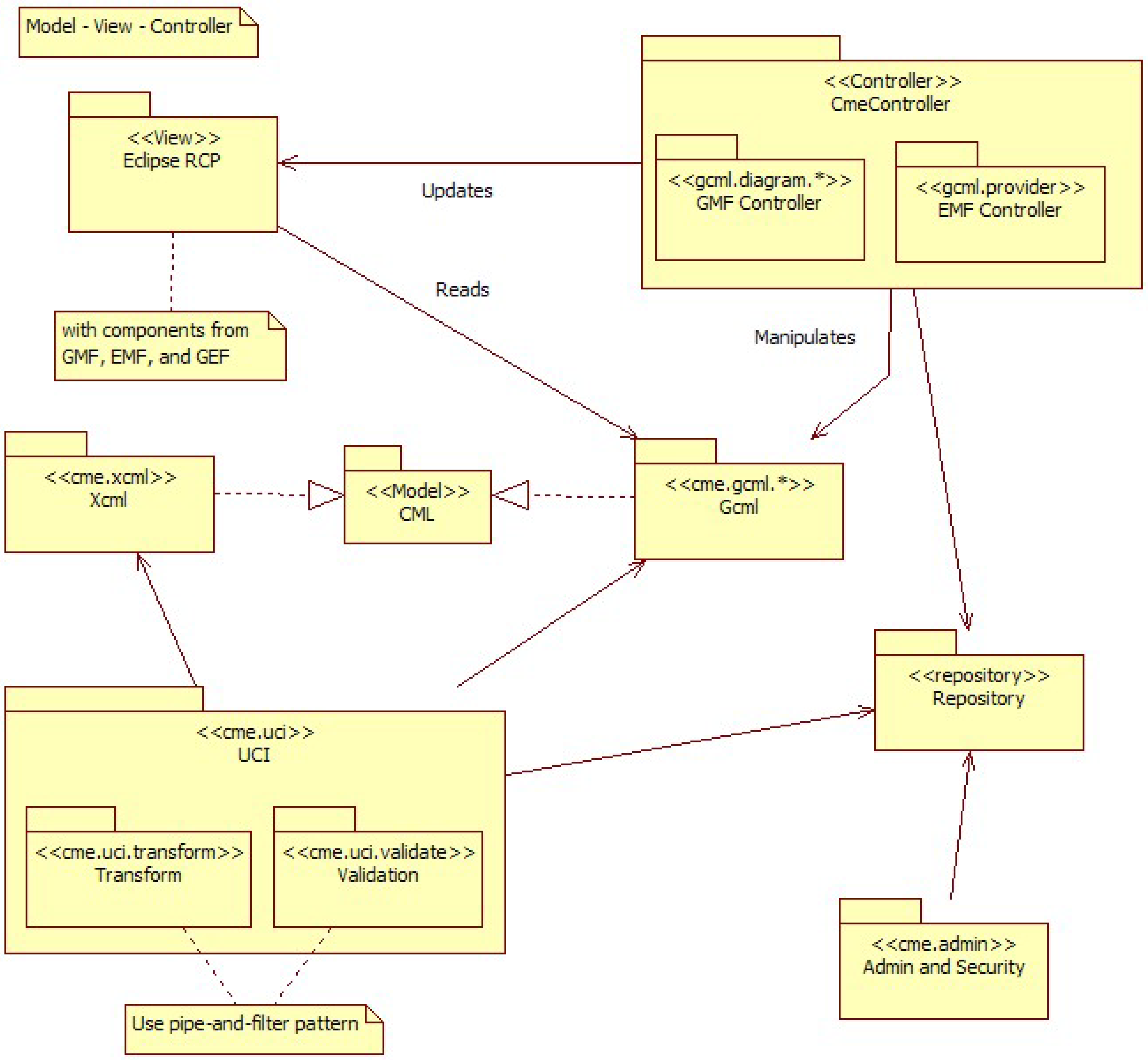


Figure 7 Package diagram

5.2. Subsystem Decomposition

CmeController (GMF controller and EMF Controller) is the control subsystem of the Modeling Environment MVC. It manipulates Gcml models, process requests from the modeler and updates the views. It also accesses the repository to save persistent data.

Eclipse RCP provides the view in conjunction with EMF, GMF and GEF components and acts as the user interface. This code is neither manually created nor generated but exists as an extensible platform that the generated code can use.

UCI (Transformation and Validation) implements two way transformation of GCML and XCML and validation of both types of models through a pipe and filter pattern. It also accesses the repository to save persistent data. In future implementation it would provide the interface to access the CVM.

Gcml object model is the realization of the CML metamodels as GCML.

Xcml object model is the realization of the CML metamodels as XCML.

Repository contains all logic to interact with the data store that holds all persistent data.

Admin and Security contains all code relevant to administrative and security functions. It also accesses the repository.

5.3. Hardware and Software Mapping

All components depend on Java 1.6

CmeController (GMF controller and EMF Controller) runs on the local computer on the Eclipse RCP (depend on EMF and GMF).

Eclipse RCP runs on the local computer (depend on EMF and GMF).

Gcml object model must be available wherever UCI runs and also for the Modeling Environment.

UCI (Transformation and Validation) this component is independent of the Eclipse RCP so it can either run on the local machine or an application server with minimal modifications.

Xcml object model must be available wherever UCI runs.

Repository does not depend on other components and can run locally, or an application server with minimal modifications (it can further be adapted to access a database server).

Admin and Security contains all code relevant to administrative and security functions; it runs on the local machine.

5.4. Persistent Data Management

The diagrams in this sections show table definitions for the main entities in the CME system. Although they are shown as SQL tables the actual implementation of the Repository is agnostic of the data store used. Other potions of store can be XML files and text files. The implementation for this prototype uses XML files in a local repository on the User's file system.

Persistent Data for Gcml

Form:

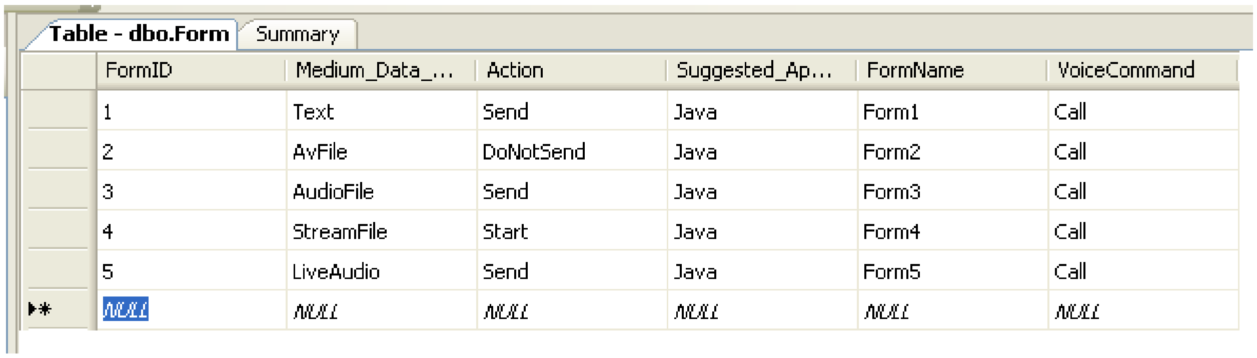


Table - dbo.Form | Summary

	FormID	Medium_Data_...	Action	Suggested_Ap...	FormName	VoiceCommand
	1	Text	Send	Java	Form1	Call
	2	AvFile	DoNotSend	Java	Form2	Call
	3	AudioFile	Send	Java	Form3	Call
	4	StreamFile	Start	Java	Form4	Call
	5	LiveAudio	Send	Java	Form5	Call
▸*	NULL	NULL	NULL	NULL	NULL	NULL

Figure 8 Persistent data for GCML, Form

Person:

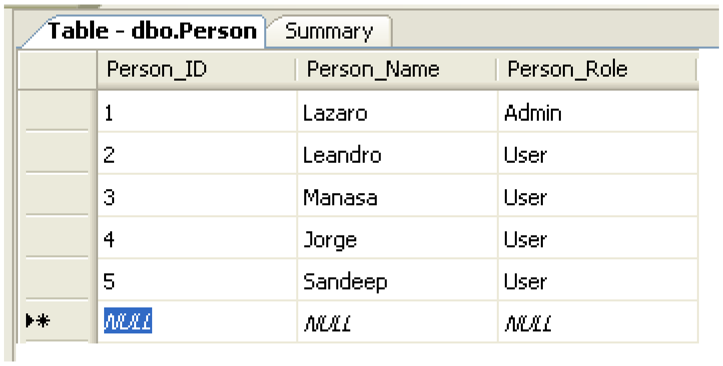


Table - dbo.Person | Summary

	Person_ID	Person_Name	Person_Role
	1	Lazaro	Admin
	2	Leandro	User
	3	Manasa	User
	4	Jorge	User
	5	Sandeep	User
▸*	NULL	NULL	NULL

Figure 9 Persistent Data for GCML, Person

Action:

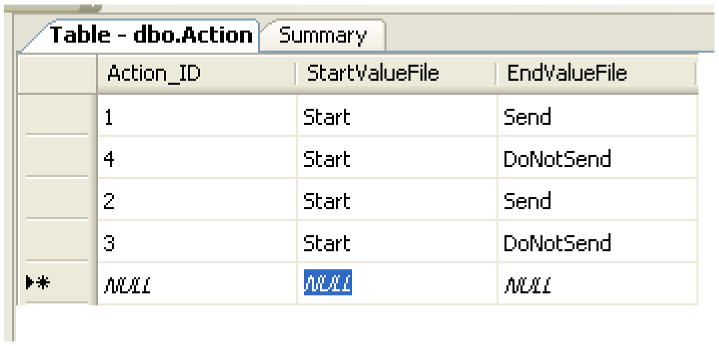


Table - dbo.Action | Summary

	Action_ID	StartValueFile	EndValueFile
	1	Start	Send
	4	Start	DoNotSend
	2	Start	Send
	3	Start	DoNotSend
▸*	NULL	NULL	NULL

Figure 10 Action

Capability Type:

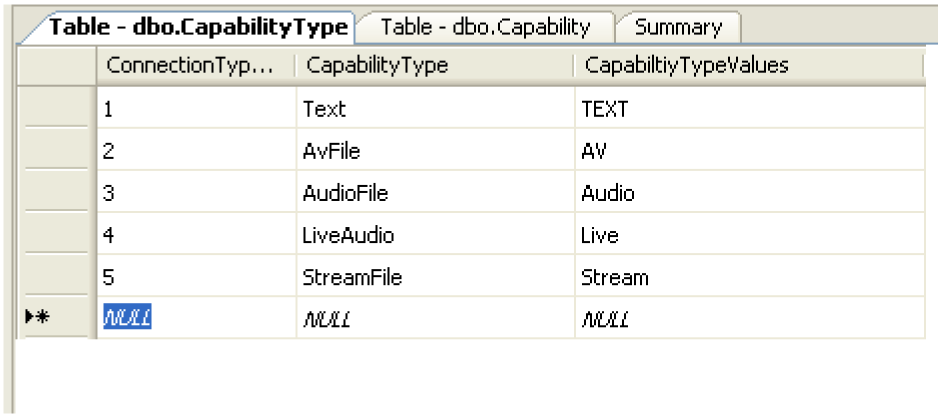


Table - dbo.CapabilityType | Table - dbo.Capability | Summary

	ConnectionTyp...	CapabilityType	CapabiltiyTypeValues
	1	Text	TEXT
	2	AvFile	AV
	3	AudioFile	Audio
	4	LiveAudio	Live
	5	StreamFile	Stream
▸*	NULL	NULL	NULL

Figure 11 CapabilityType

Device:

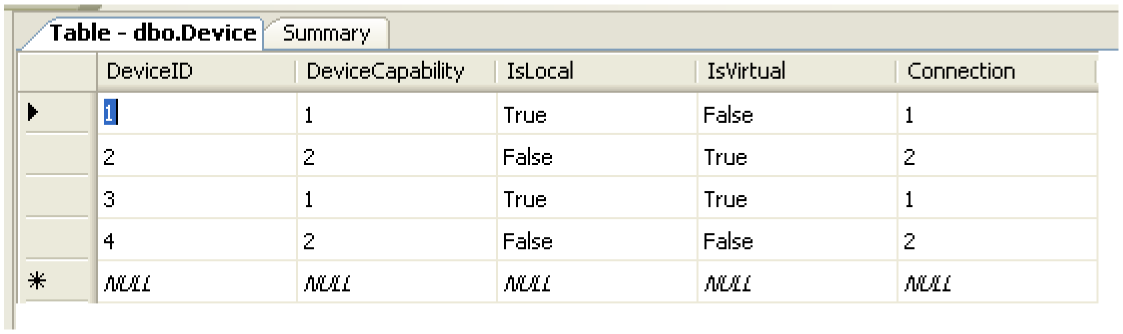


Table - dbo.Device | Summary

	DeviceID	DeviceCapability	IsLocal	IsVirtual	Connection
▸	1	1	True	False	1
	2	2	False	True	2
	3	1	True	True	1
	4	2	False	False	2
*	NULL	NULL	NULL	NULL	NULL

Figure 12 Devic

User

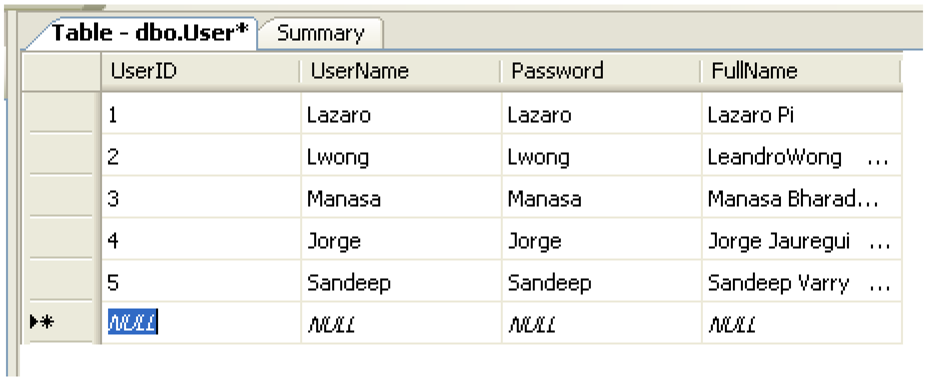


Table - dbo.User* | Summary

	UserID	UserName	Password	FullName
	1	Lazaro	Lazaro	Lazaro Pi
	2	Lwong	Lwong	LeandroWong ...
	3	Manasa	Manasa	Manasa Bharad...
	4	Jorge	Jorge	Jorge Jauregui ...
	5	Sandeep	Sandeep	Sandeep Varry ...
▸*	NULL	NULL	NULL	NULL

Figure 13 User

6.Object Design

After the discussion of system decomposition in the software architecture step, the object design for each subsystem is presented in this chapter focusing on both the static and dynamic views.

Each subsystem is divided into classes. We give the minimum class diagram for the subsystems in the overview subsection. The sequence diagrams show the interactions between objects and actors. These actions are described in the appropriate use cases. The statechart diagrams show dynamic behavior and state transitions of the main subsystems. Finally, we give the detailed classes design and explain the purpose of each class. The classes are grouped under packages representing the software architecture of the system.

6.1. Overview

Minimal Class Diagrams

CmeController

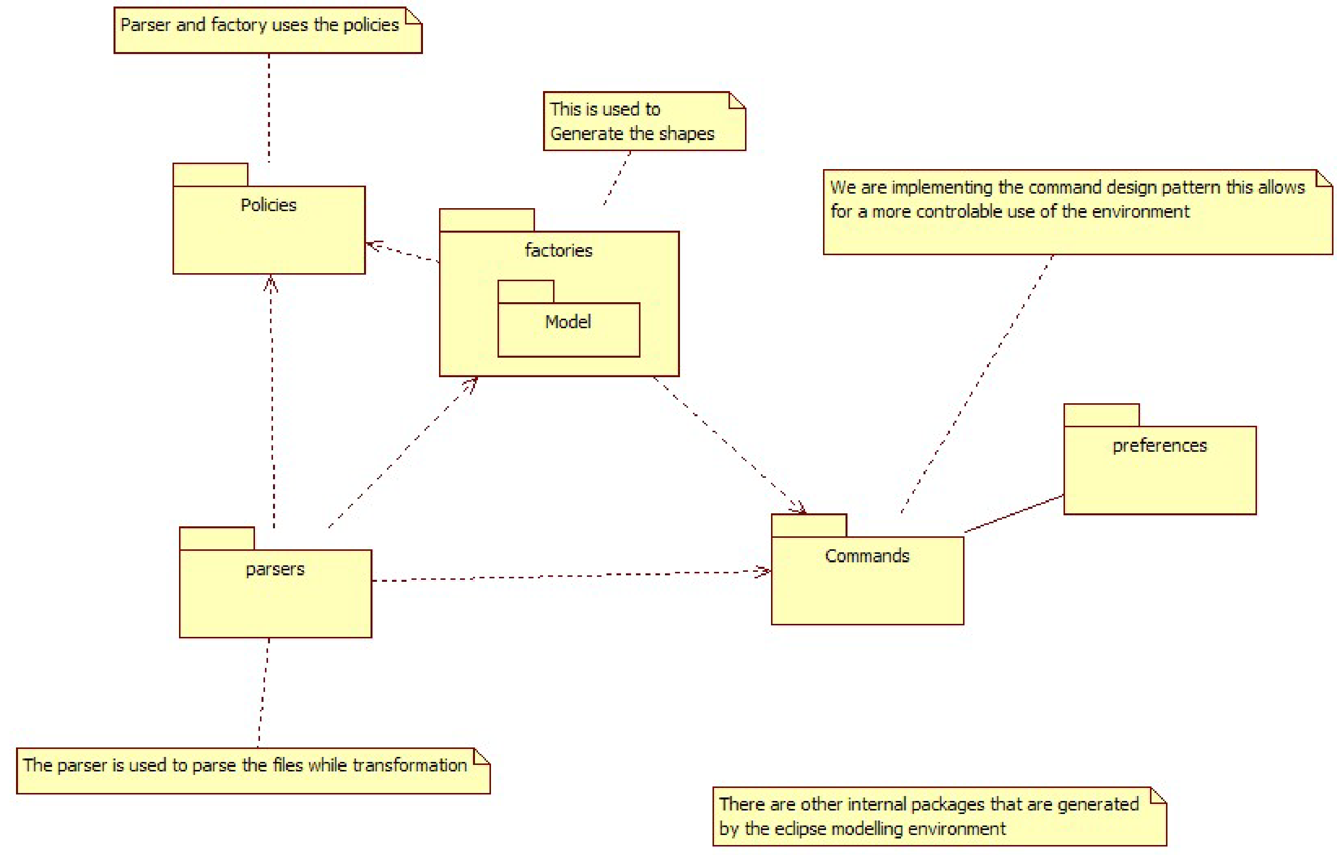
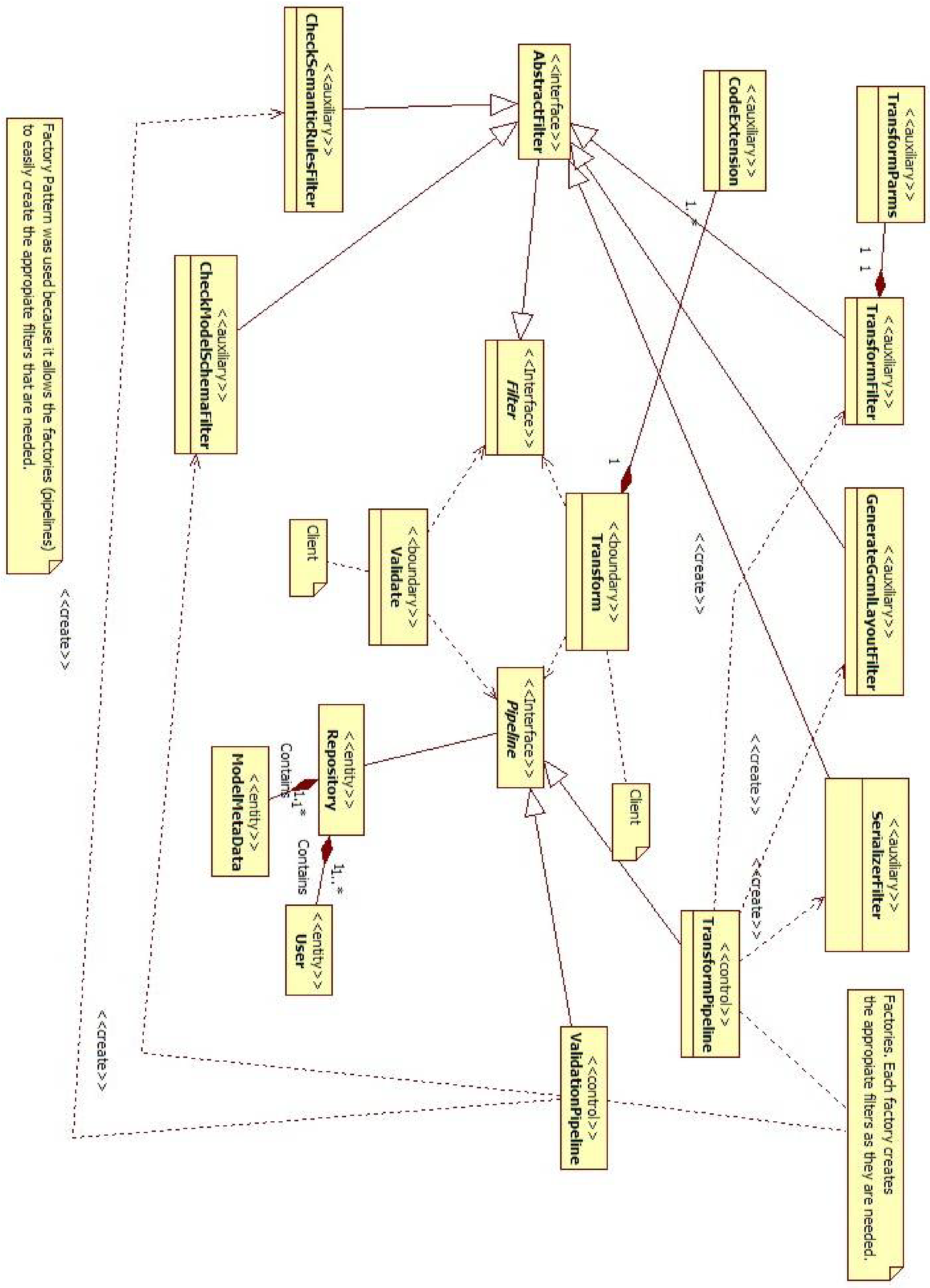

Parser and factory uses the policies
This is used to Generate the shapes
We are implementing the command design pattern this allows for a more controlable use of the environment
Policies
factories
Model
preferences
parsers
Commands
The parser is used to parse the files while transformation
There are other internal packages that are generated by the eclipse modelling environment

Figure 14 Minimal Class Diagral, CmeController

UCI and Repository



<<auxiliary>> TransformParms
<<auxiliary>> CodeExtension
<<interface>> AbstractFilter
<<auxiliary>> CheckSemanticRulesFilter
<<auxiliary>> CheckModelSchemaFilter
<<Interface>> Filter
<<auxiliary>> TransformFilter
<<boundary>> Transform
<<boundary>> Validate
Client
<<auxiliary>> GenerateGcmlLayoutFilter
<<Interface>> Pipeline
<<entity>> Repository
<<entity>> ModelMetaData
<<entity>> User
Contains
Contains
1..1*
1..*
1..*
1
1 1
<<auxiliary>> SerializerFilter
<<control>> TransformPipeline
<<control>> ValidationPipeline
<<create>>
<<create>>
<<create>>
<<create>>
<<create>>
Client
Factory Pattern was used because it allows the factories (pipelines) to easily create the appropriate filters that are needed.
Factories. Each factory creates the appropiate filters as they are needed.

Figure 15 Minimal Class Diagram, UCI and repository

Comments

Briefly describe the purpose of having certain classes in certain subsystems

Gcml

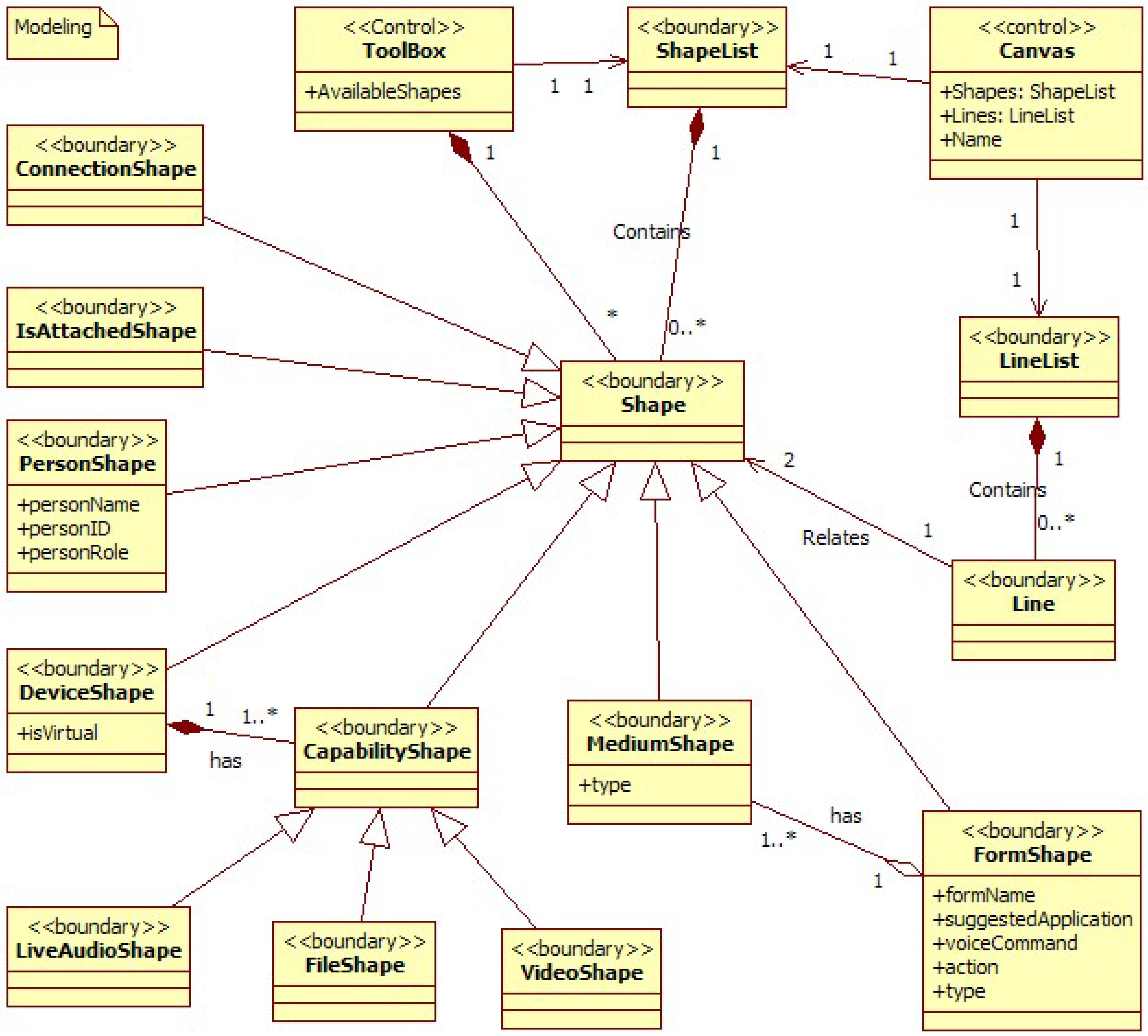


Figure 16 Minimal Class Diagral, GCML

Xcml

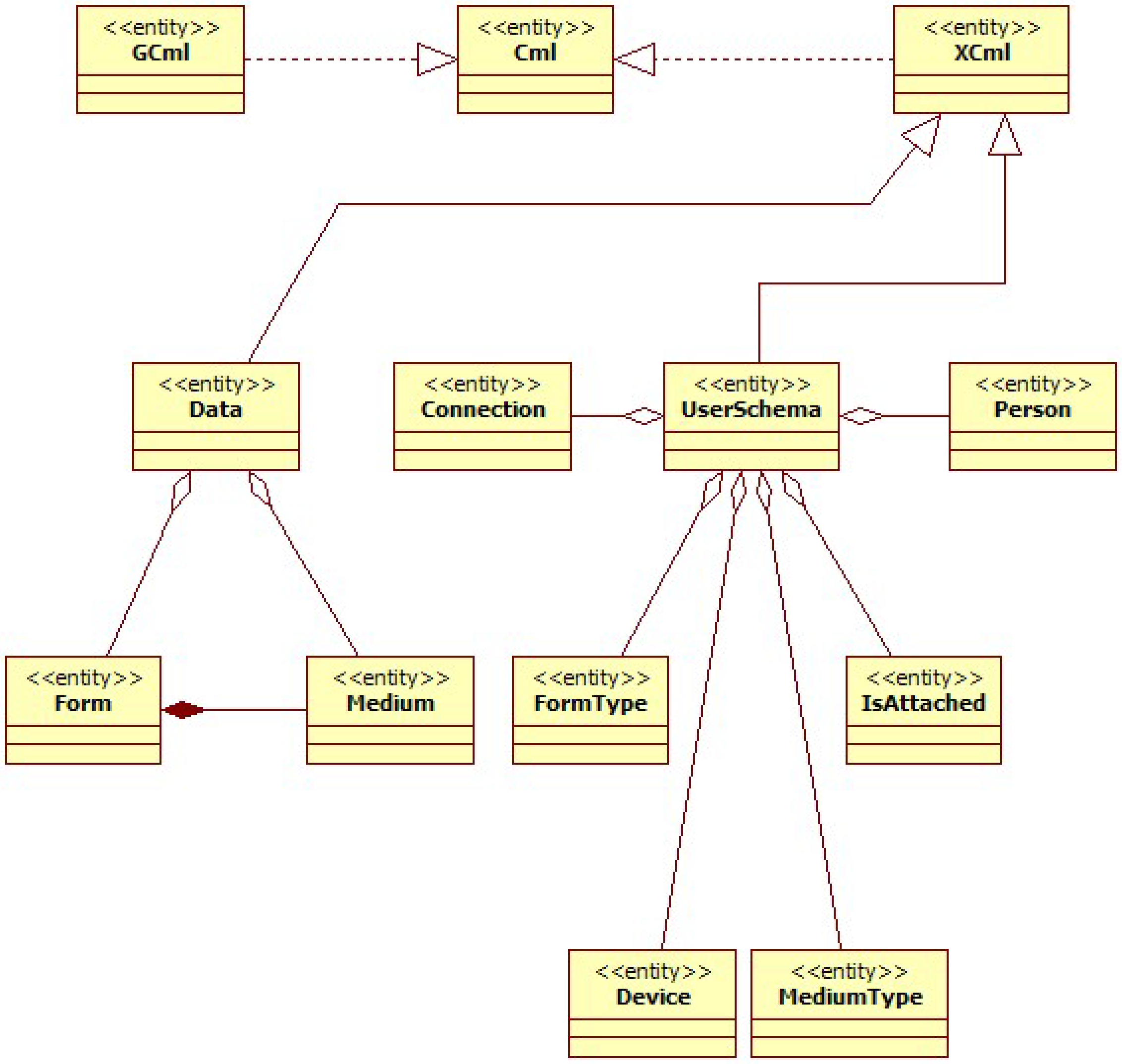
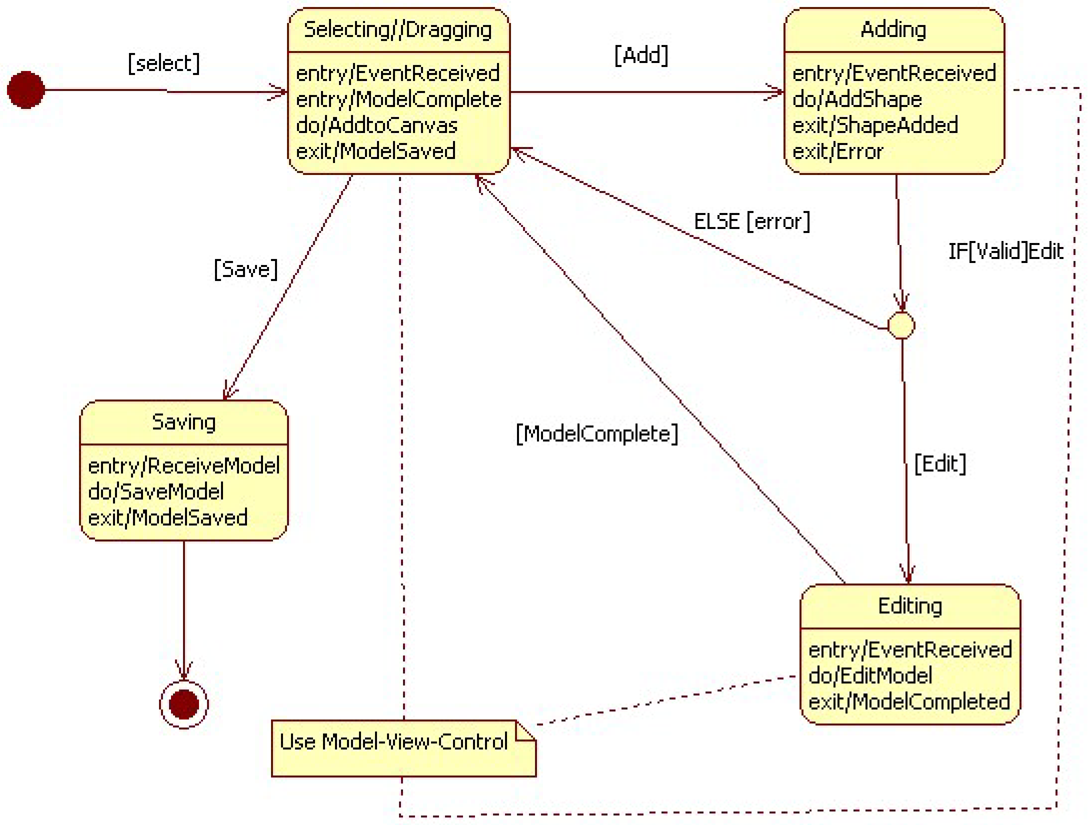


Figure 17 Minimal Class Diagram XCML

6.2. State Machine

Model Creation Statechart



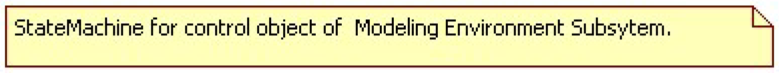

StateMachine for control object of Modeling Environment Subsytem.

Figure 18 Model Creation StateChart

The above figure is the state chart diagram representing the dynamic behavior for the control part in the Modeling Environment subsystem (referred as CmeController). The chart highlights the main functions of the Modeling Environment, loading, saving, and editing models.

Model Transformation Statechart

The figure below corresponds to the model transformation functionalities (part of the UCI subsystem). It can be seen that conversion between GCML and XCML is achieved using a pipe and filter pattern. The pipelines for these functions are initialized through factory methods. The UCI component implements validation using a similar approach.

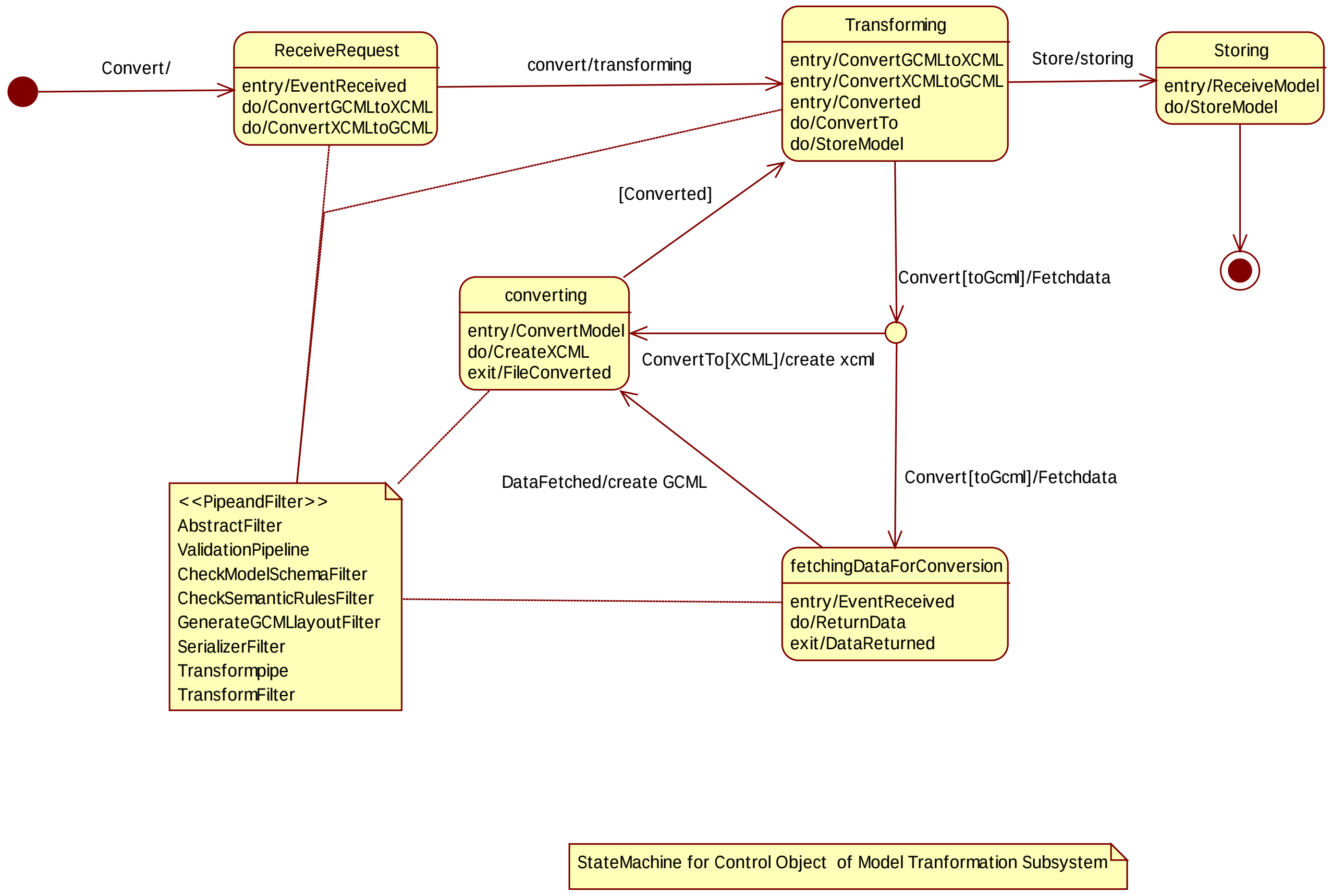


Figure 19 Model Transformation Statechart

Repository Statechart

The next figure corresponds to the Repository subsystem. The main functionality of this subsystem is to act a a central store for the models, their metadata. It also stores information for the security and administration functions of the system. Main functions allow retrieving and saving data into an abstracted data store. This will allow the introduction of different data stores later (File Servers, Database Servers, etc).

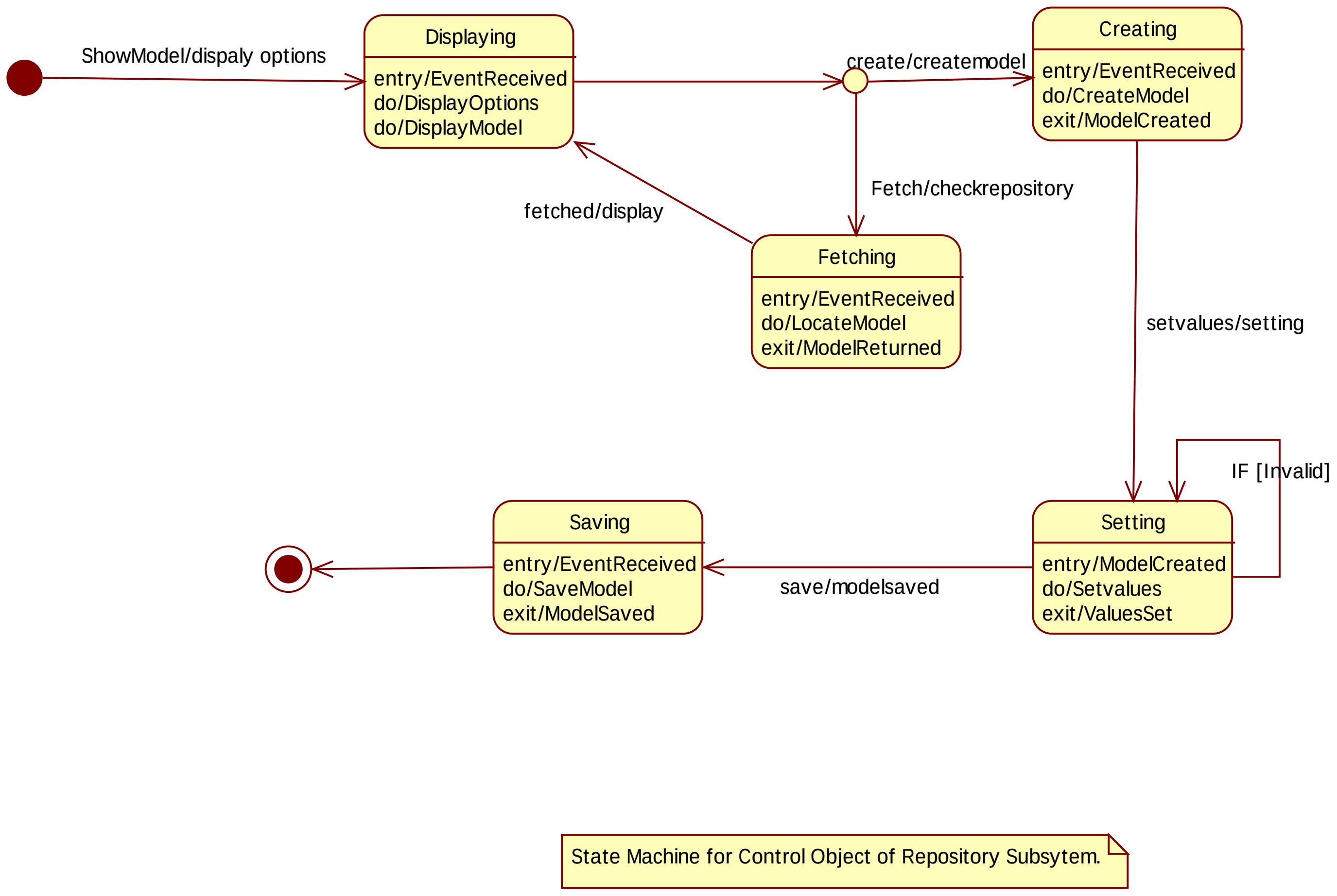


Figure 20 Repository Statechart

Administration and Security Statechart

The next figure corresponds to the Administration and Security subsystem. We are showing the functions for authentication of users and how this relates to the available functionalities and controls based on the type of user, wither administrator or modeler. Other functions described in the identified use cases are not shown (for example, creation and deletion of user accounts).

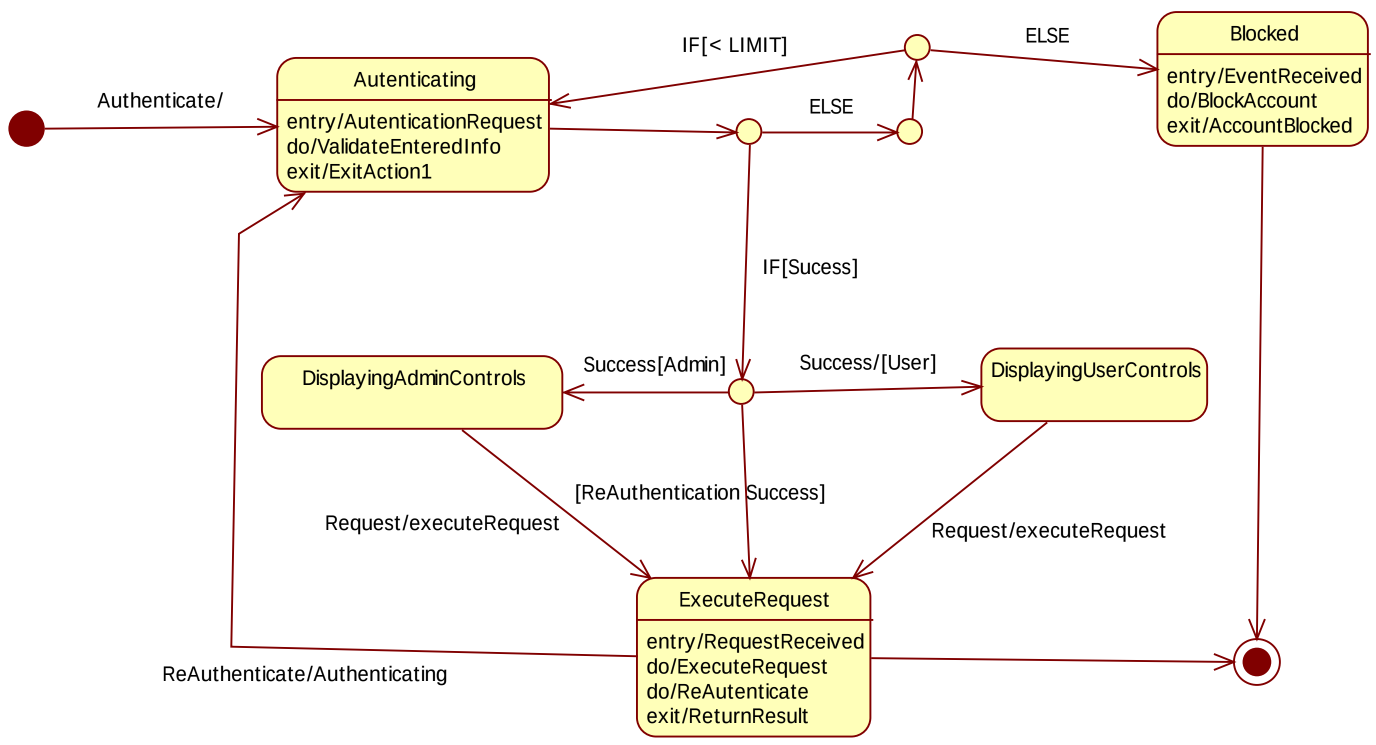


Figure 21 Administration and Security Statechart

6.3. Object Interaction (Sequence Diagrams)

6.3.1. Open Exiting GCML Model

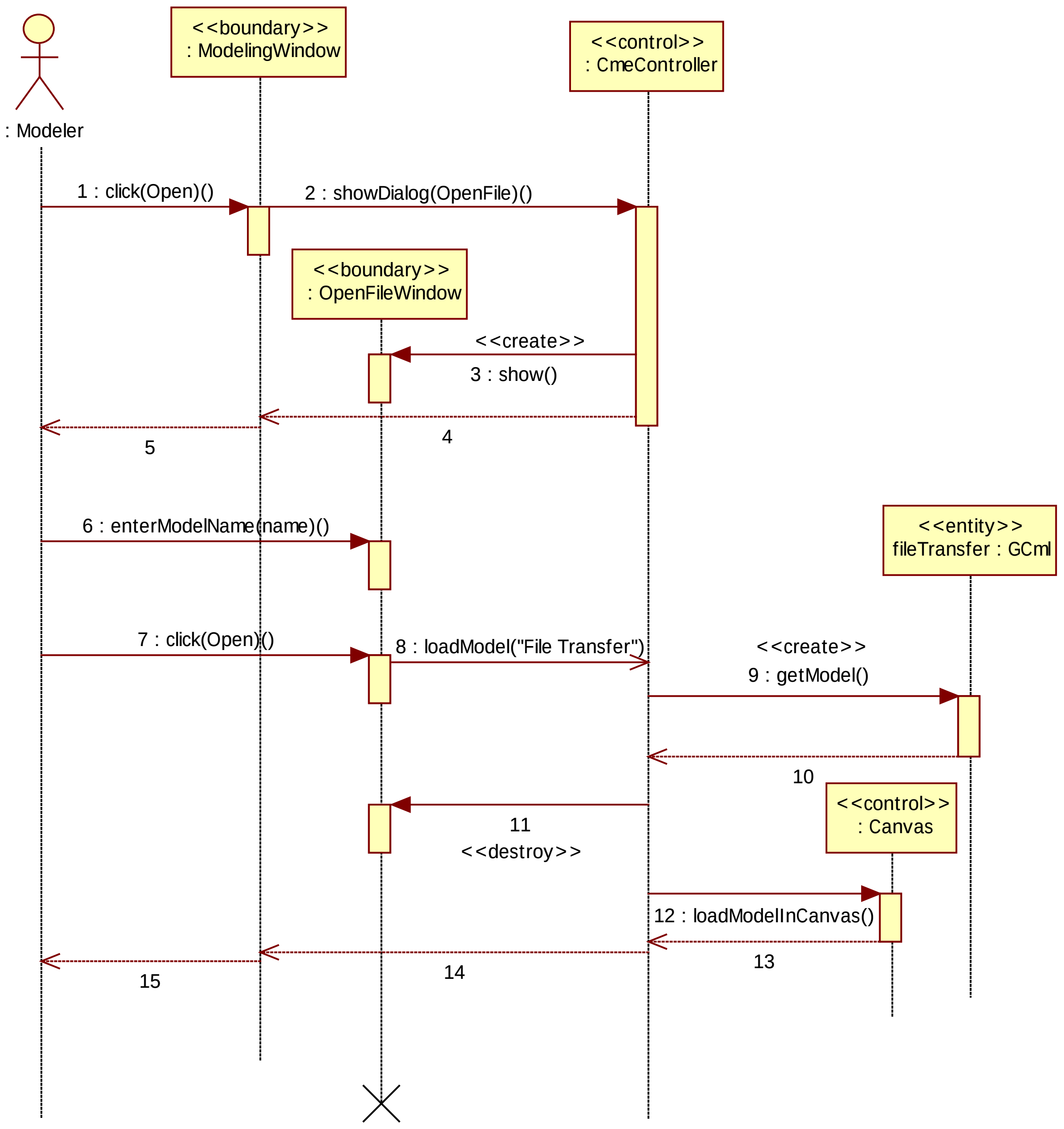
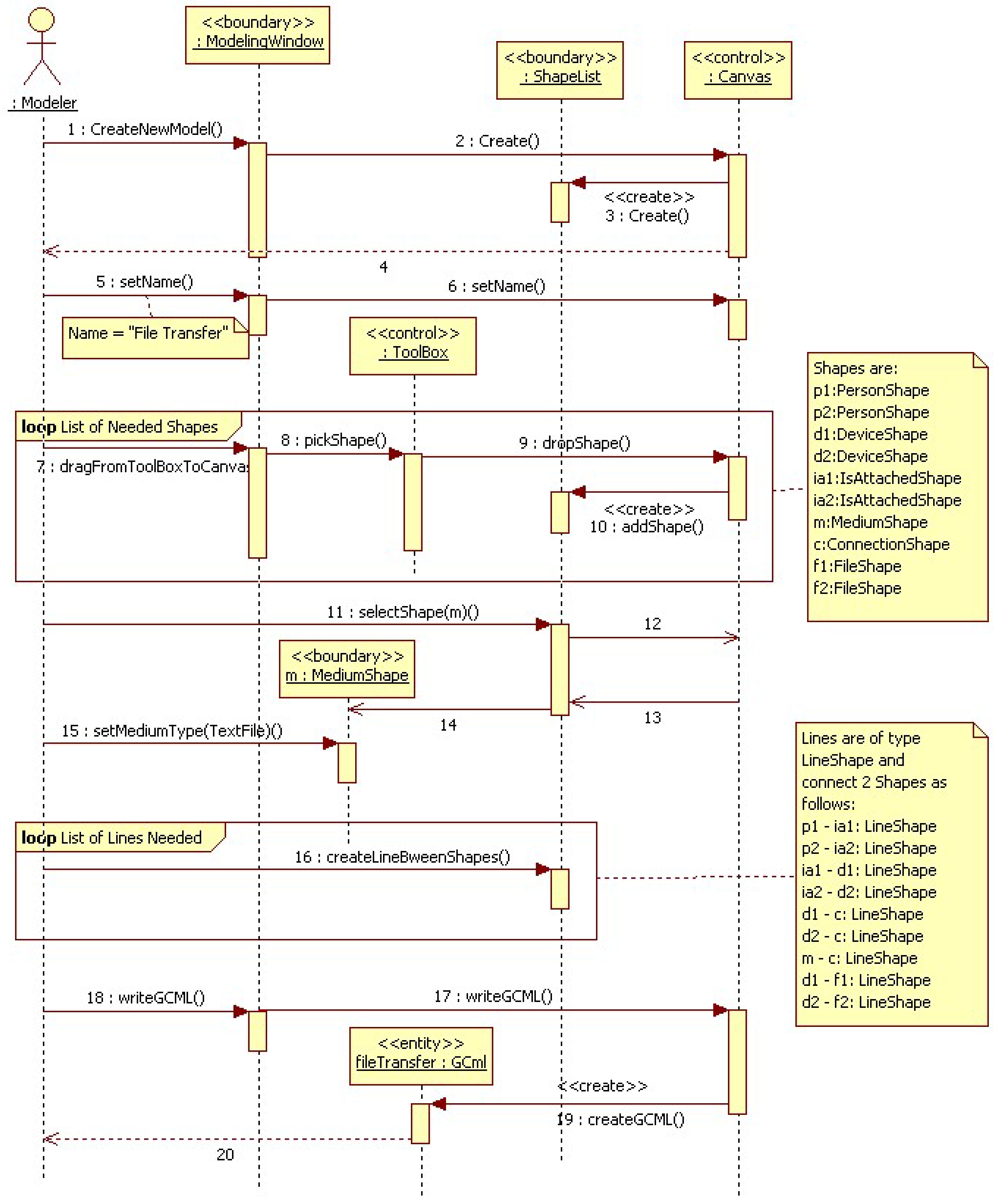

: Modeler
<<boundary>> : ModelingWindow
<<control>> : CmeController
<<boundary>> : OpenFileWindow
<<entity>> fileTransfer : GCml
<<control>> : Canvas
1 : click(Open)()
2 : showDialog(OpenFile)()
<<create>>
3 : show()
4
5
6 : enterModelName(name)()
7 : click(Open)()
8 : loadModel("File Transfer")
<<create>>
9 : getModel()
10
11
<<destroy>>
12 : loadModelInCanvas()
13
14
15

Figure 22 Sequence Diagram , open gcml

6.3.2. Create File Transfer Model



: Modeler
<<boundary>> : ModelingWindow
<<boundary>> : ShapeList
<<control>> : Canvas
1 : CreateNewModel()
2 : Create()
<<create>> 3 : Create()
4
5 : setName()
6 : setName()
Name = "File Transfer"
<<control>> : ToolBox
loop List of Needed Shapes
7 : dragFromToolBoxToCanvas
8 : pickShape()
9 : dropShape()
<<create>> 10 : addShape()
Shapes are: p1:PersonShape p2:PersonShape d1:DeviceShape d2:DeviceShape ia1:IsAttachedShape ia2:IsAttachedShape m:MediumShape c:ConnectionShape f1:FileShape f2:FileShape
11 : selectShape(m)()
12
<<boundary>> m : MediumShape
13
14
15 : setMediumType(TextFile)()
Lines are of type LineShape and connect 2 Shapes as follows: p1 - ia1: LineShape p2 - ia2: LineShape ia1 - d1: LineShape ia2 - d2: LineShape d1 - c: LineShape d2 - c: LineShape m - c: LineShape d1 - f1: LineShape d2 - f2: LineShape
loop List of Lines Needed
16 : createLineBweenShapes()
18 : writeGCML()
17 : writeGCML()
<<entity>> fileTransfer : GCml
<<create>> 19 : createGCML()
20

Figure 23 Sequence Diagram, Create File Transfer Model

6.3.3. Convert Model from GCML to XCML

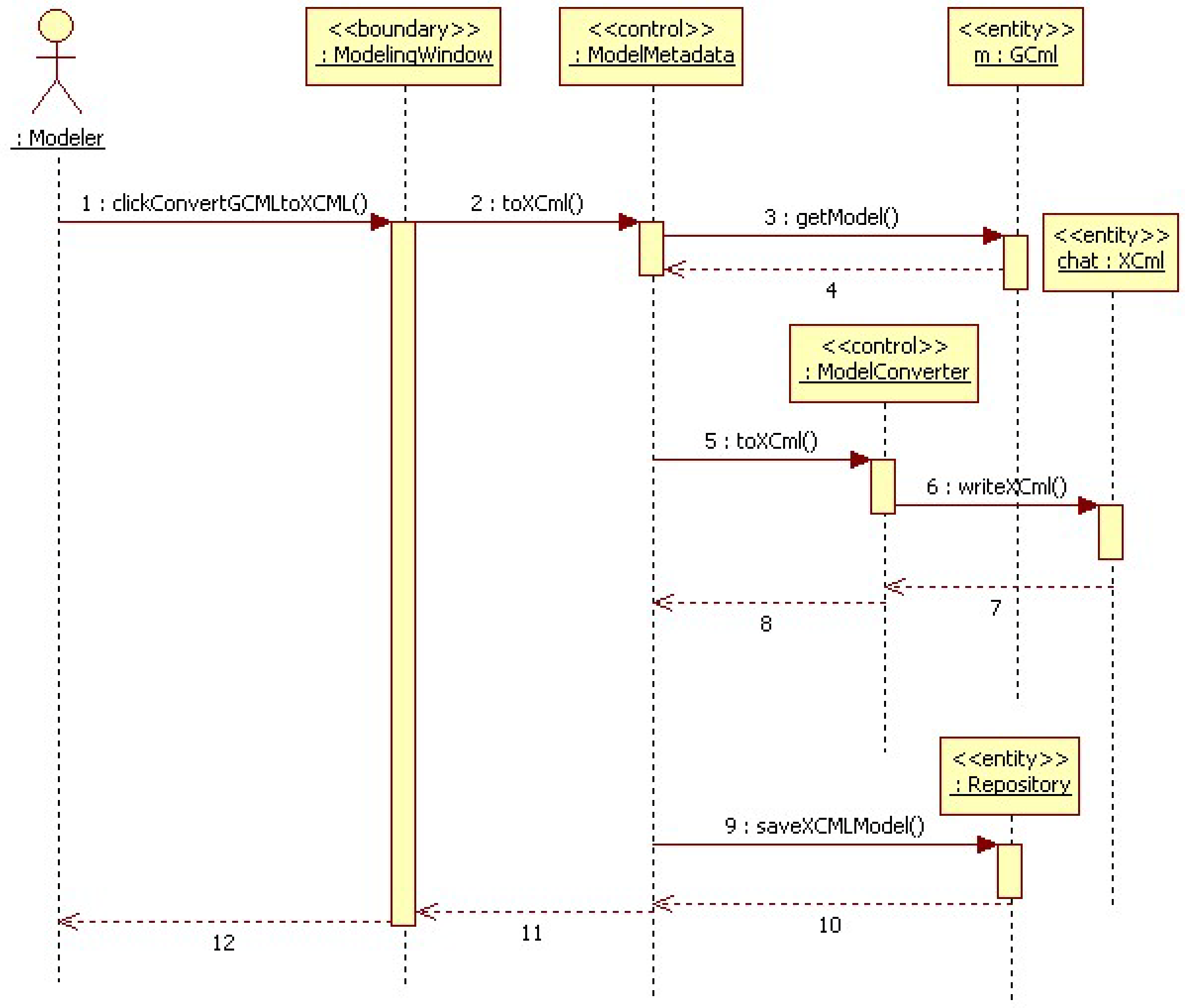


Figure 24 Sequence Diagram, Convert GCML to XCML

6.3.4. Convert Model from XCML to GCML

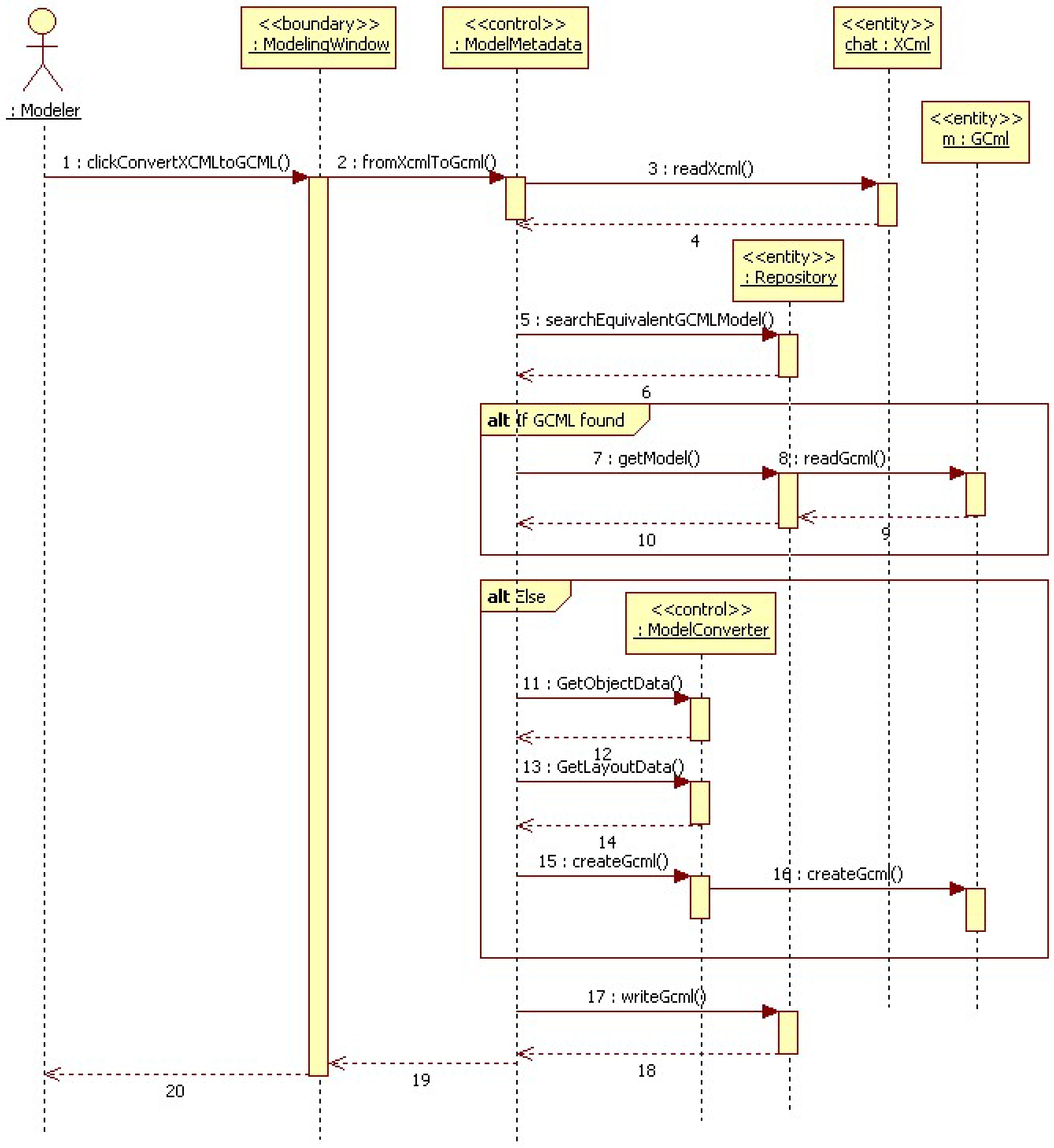
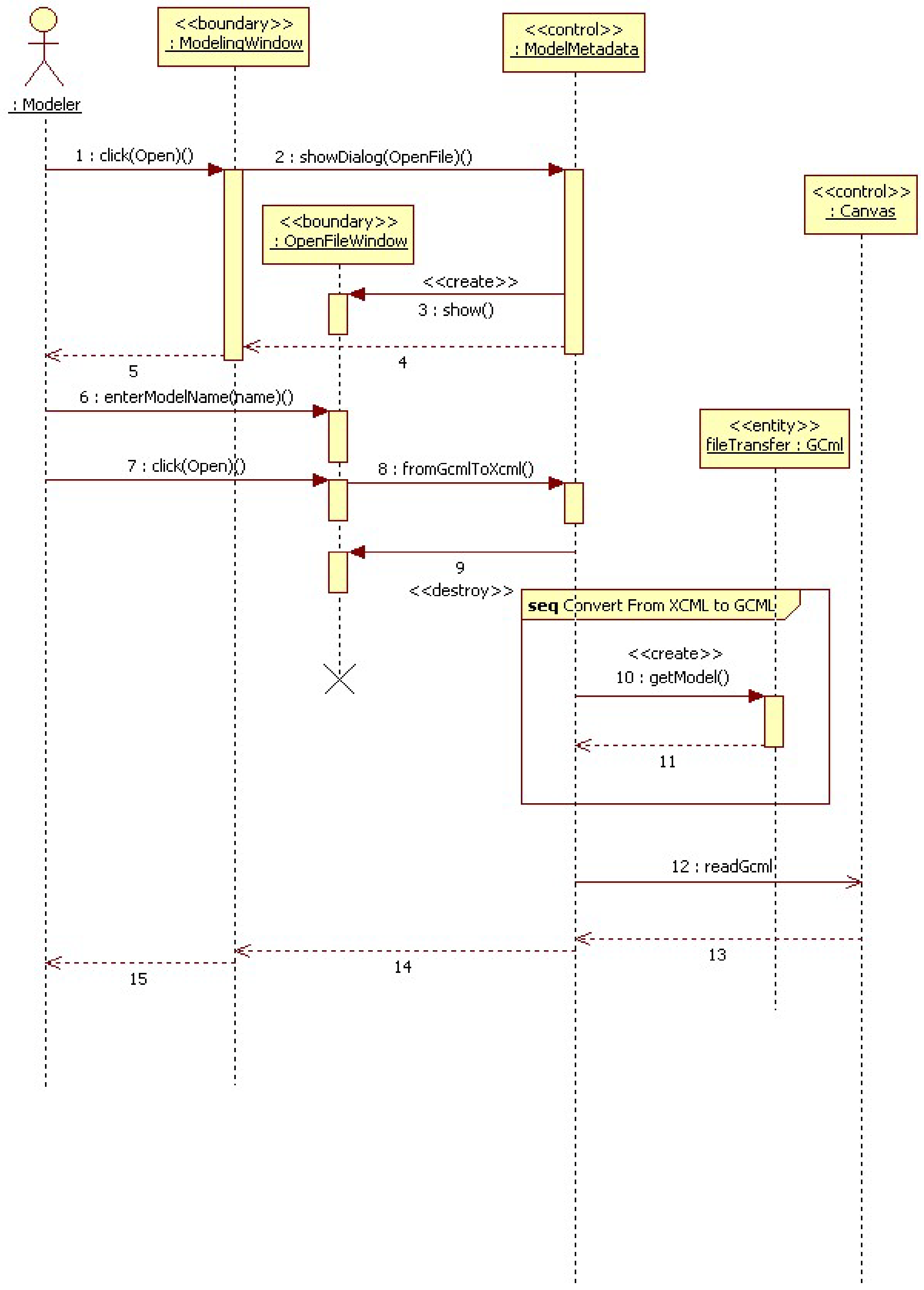


Figure 25 Sequence Diagram, Convert XCML to GCML

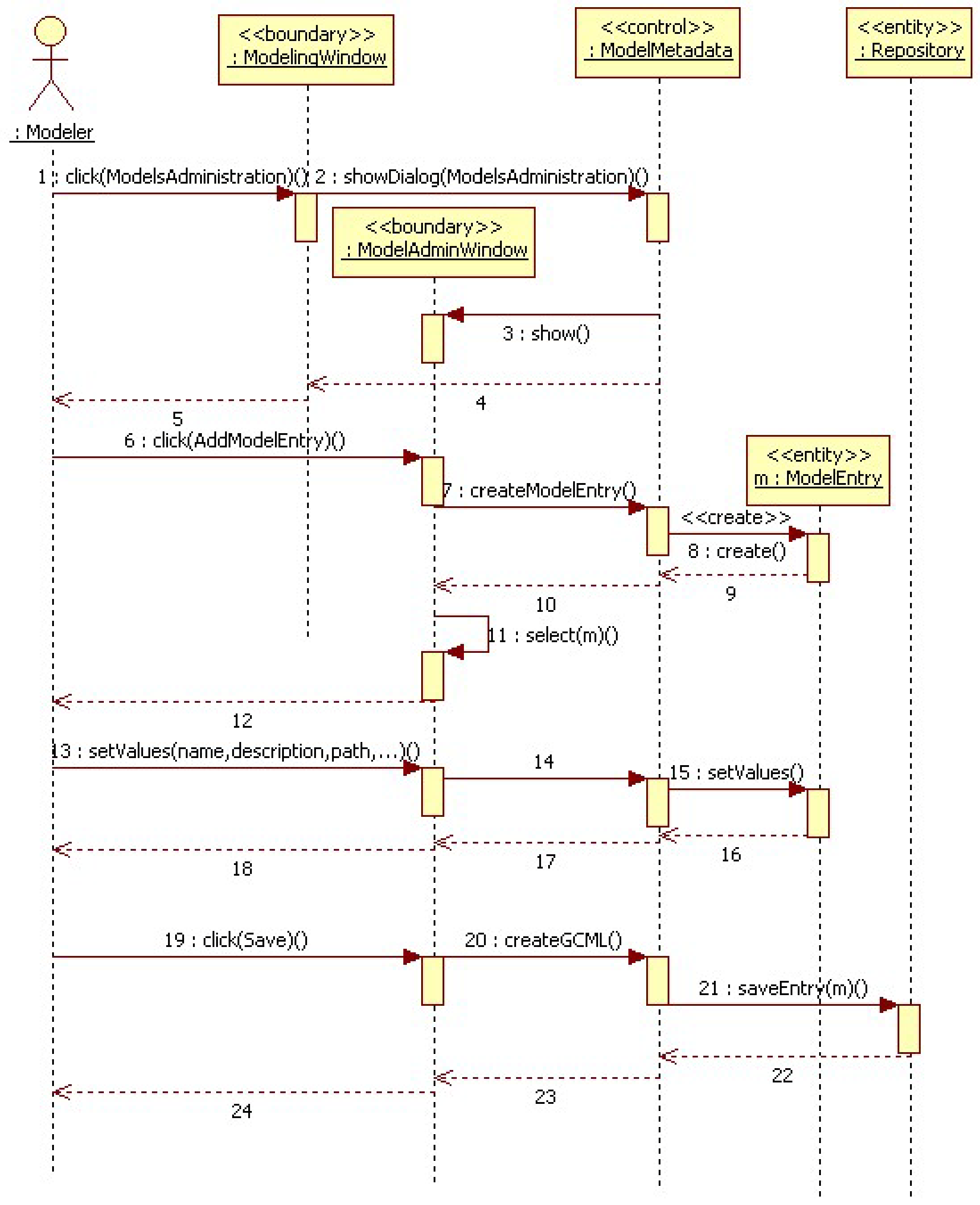
6.3.5. Import Model from XCML



: Modeler
<<boundary>>
: ModelingWindow
<<control>>
: ModelMetadata
<<control>>
: Canvas
<<boundary>>
: OpenFileWindow
1 : click(Open)()
2 : showDialog(OpenFile)()
<<create>>
3 : show()
4
5
6 : enterModelName(name)()
7 : click(Open)()
8 : fromGcmlToXcml()
<<entity>>
fileTransfer : GCml
9
<<destroy>>
seq Convert From XCML to GCML
<<create>>
10 : getModel()
11
12 : readGcml
13
14
15

Figure 26 Sequence Diagram, Import form XCML

6.3.6. Add Model to Repository



<<boundary>> : ModelingWindow
<<control>> : ModelMetadata
<<entity>> : Repository
: Modeler
1 : click(ModelsAdministration)()
2 : showDialog(ModelsAdministration)()
<<boundary>> : ModelAdminWindow
3 : show()
4
5
6 : click(AddModelEntry)()
<<entity>> m : ModelEntry
7 : createModelEntry()
<<create>>
8 : create()
9
10
11 : select(m)()
12
13 : setValues(name,description,path,...)()
14
15 : setValues()
16
17
18
19 : click(Save)()
20 : createGCML()
21 : saveEntry(m)()
22
23
24

Figure 27 Sequence diagram, Add Model to Repository

6.3.7. Edit Model Metadata

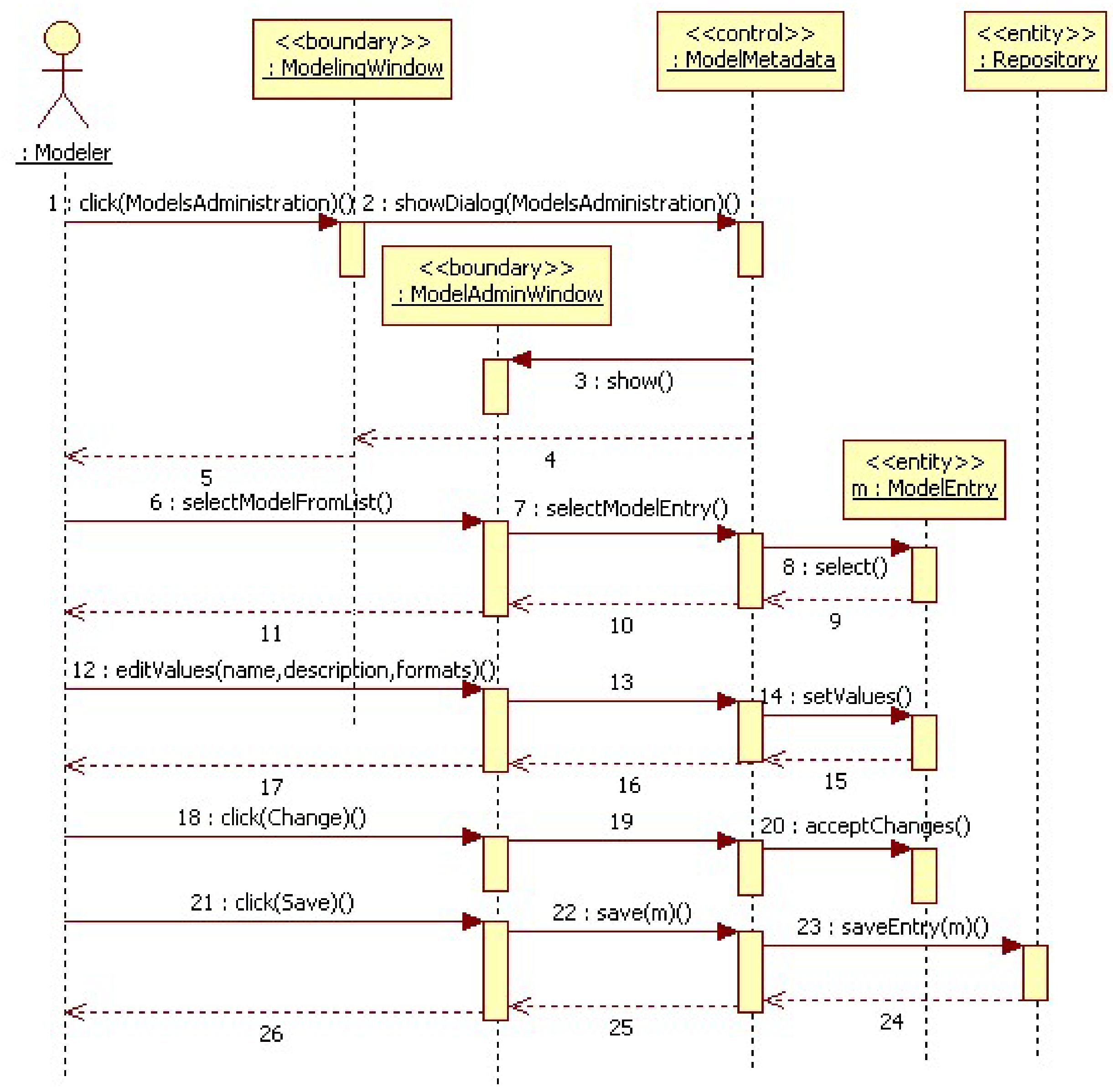


Figure 28 Sequence Diagram, Edit model Metadata

6.3.8. Validate Model

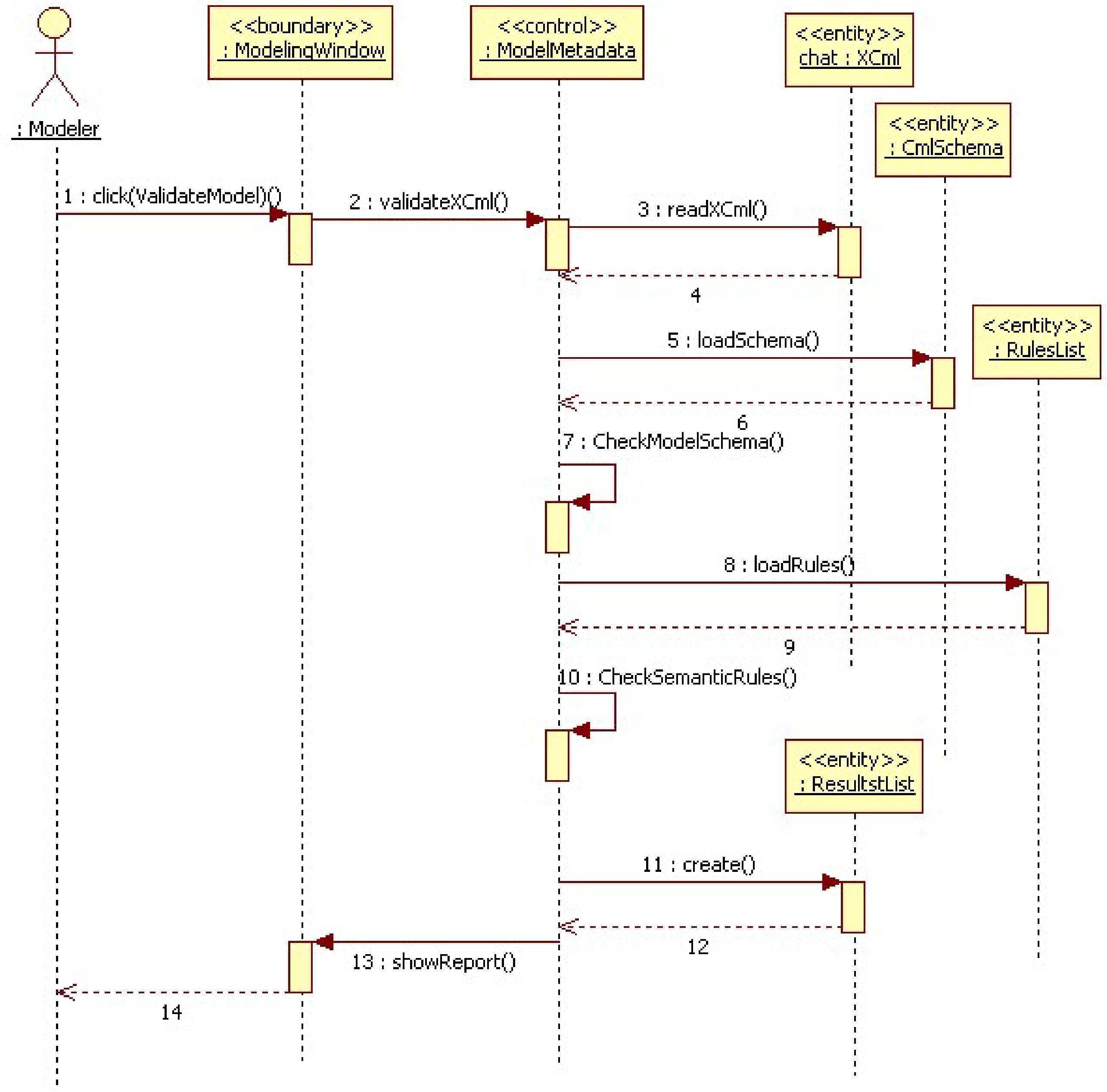


Figure 29 Sequence Diagram, Validate Model

6.4. Detailed Class Design

Design Patterns Used

Singleton Pattern

The Singleton Pattern was used in our application to manage the creation of various classes, such as the Repository class and the current logged in user (which initialized when calling the login method). Any calls to these classes should be done to the same instance; that way the values of settings (such as Repository paths and user rights) are preserved through the entire application.
Lazy initialization was also used on those classes to postpone their creation until they were actually needed.

Factory Method Pattern

The Pipeline classes of the UCI (Validate and Transform) component uses private constructors and factory methods to control access to the pipeline initialization; this allows flexibility in changing the order, code extensions, and behaviors of the filters (this include suspend the execution to be resumed later, etc.), and also the way they are read (from ".config" file, from code, or from central Database) without affecting the client code that needs to execute the Pipeline.

Adapter Pattern

Since there are 2 variations of CML (Gcml and Xcml), the UCI package acts as an adapter between the CVM which only understands Xcml and the Modeling Environment which only understands Gcml. Furthermore, the use of CML isolates the client applications from the underlying communication API's (whichever they may be).

Observer Pattern and Command Pattern

This pattern is used in the CmeController to listen to events from the modeler. Many of the functions to edit the Gcml model are implemented as command classes in the cme.diagram project (for example, the PersonCreateCommand class).

Class Descriptions

CmeController (see Appendix C)

Command package: Encapsulates the user requests in the modeling environment.
Factory package: It manages the creation of the shapes in the modeling environment. Each shape is defined in the CML subsystem and accessed through the repository.
Parser package: Is use to parse the information from each shapes.

Repository (see Appendix C)

ModelMetaData: Meta Data representation of CVM.
User: Contains the user information.
Repository: It implements Singleton. It contains a collection of Users and the MetaData.

UCI Engine (see Appendix C)

Transform: It converts a file from Xcml to Gcml or viceversa.

- Precondition: Validcml(Afile) = True.
- Postcondition: fromXcmlToGcml = Gcml file
- Postcondition: fromGcmlToXcml = Xcml file

Validation: It performs the validation to the files being uploaded. Two main filters are CheckModelSchemaFilter and CheckSemanticRulesFilter.

- Precondition: Validcml(Afile) = True.
- Postcondition: ValidSemantic(Afile) = True.
- Postcondition: ValidModel (Afile) = True.

Pipeline: Represents the Transformations as a linear pipeline where all the steps are executed in order and the input of one step is the output of the previous step. The last output is returned to the client that called the execute method.
TransformPipeline: It inherits from pipeline. It does the transformation by applying the corresponding filters to each xml tag in the file.
ValidationPipeline: It inherits from pipeline. It performs the validation by selecting the appropiate filter to each xml tag in a *Pipe & Filter* behavior.
Filter: Represents one step in the pipeline of transformations. Interface
AbstractFilter: A generic implementation of the Filter Interface. The Transform and Validation filters inherit from this class.
TransformFilter: It inherits from Abstract Filters. Use by the Transform pipeline.
Serializer Filter: It inherits from Abstract Filters. Use by the Transform pipeline.
GenerateGcmlLayoutFilter: It inherits from Abstract Filters. Use by the Transform pipeline when transforming from Xcml to Gcml.
CheckModelSchemaFilter: It inherits from Abstract Filters. Use by the Validation pipeline. It verifies the association rules (MetaData) between shapes.
CheckSemanticRulesFilter: It inherits from Abstract Filters. Use by the Validation pipeline. It verifies semantic rules in the diagram.
TransformParms: Parameters to pass to the Transform method.
CodeExtension: All code extensions passed to the Transform must implement this Interface.

CML Model

Person:A person participating in the communication.
LocalPerson : Person: The local person; it can be only one per application.
RemotePerson : Person:A remote person; there can be more than one.
Medium: The medium to exchange in this connection.
File: Medium: A File to be sent through the connection used to create chat connections.
TextStream: Medium: used to create chat connections.
VoiceStream: Medium: used to create voice calls.
Connection: Represents the connection between 2 or more devices.
Device: Represents the Device used by a person to connect to another.
IsAttached: A link between persons and devices.

7.Testing Process

Testing process cover a detailed testing of the system at various levels like system level, sub system level and also unit level. The test cases presented in the systems test cases test the implemented use cases which test the basic functionality of the system and the test cases are divided into two sunny day and one rainy day for each of the use cases. The unit test cases are implemented in the Junit.

7.1. System Tests.

Test Cases for Login Log out (Security Use Case) :

Sunny day Scenario : The Jjauregui a valid user logs into the system

Test ID	TCS_1.1 Login to the System
Purpose	To test if a user can successfully log on to the system. The user cannot view the modeling environment till he enters a valid user name and password.
Preconditions	Jjauregui is registered into the system; If Jjauregui is admin then he has all the accesses privileges. If the user is a normal user, admin has already assigned the accesses privileges.
Input	• Jjauregui (administrator) goes to the Login User Screen • Jjauregui enters the Username "jjauregui" and password "superclubs". • Jjauregui hits the "Enter" Button.
Expected Output	The Modeling Environment is launched.

Sunny Day Scenario 2 : The userL1, valid user logs into the system

Test ID	TCS_1.2(Login to the System)
Purpose	To test if a user can login to the system.
Preconditions	L1 is previously added to the system by the admin and has appropriate accesses privileges.
Input	• L1 goes to the Login Screen • L1 enters the information User Name "l1" and Password "repository" • L1 hits the "Enter" button
Expected Output	The Modeling Environment is launched.

Rainy Day : The user tries to log in with valid user Name and invalid password

Test ID	TCS_1.3(Login to the System)
Purpose	To test if a user can login to the system to the system
Preconditions	L2 who is not added by the admin, tries to log in to the system.
Input	• L2 goes to the Login Screen • L2 enters the information User Name "L1" and Password "J1" • L2 hits the "Enter" button
Expected Output	The system indicates to the User an Error Message "Invalid User Name or Password , Please Try again"

Test Case for Opening an Existing GCML Model

Sunny Day : The User selects the existing model from the repository. The model comes up on the canvas.

Test ID	TCS_2.1(Open an existing GCML model)
Purpose	To test if a user “L1” can open an existing GCML model from the repository.
Preconditions	The user “L1” is logged into the application. The system has identified the user to be a valid User.
Input	• L1 clicks File -> Open and • Browses for the GCML file from “c://cme/Example/GCML1.gcml_diagram”(relative path) . • L1 clicks on “Open”.
Expected Output	The GCML1.gcml_diagram is displayed on the canvas

Sunny day 2 : The User tries to open a model when another model is already open and is being display on the Canvas.

Test ID	TCS_2.2(Open an existing GCML model when another model is open already)
Purpose	To test if L1 who is already logged into the system, can open another valid model present in the repository when another model is already open on the canvas
Preconditions	The user ("L1") is already logged on to the system.
Input	• L1 click File -> Open and • Browses for the GCML file from "c:/../cme/Example/GCML2.gcml_diagram"(relative path) . • L1 clicks on "Open".
Expected Output	A new tab is opened and the new model GCML2.gcml_diagram is displayed on the separate tab.

Rainy Day Scenario: The user tries to open a model that has been opened or edited in another modeling environment, hence no longer has an extension of .GCML

Test ID	TCS_2.3(Open an GCML model that no longer complies with the system)
Purpose	To test if L1 can open an invalid GCML .
Preconditions	The user ("L1") is already logged on to the system.
Input	• L1 click File -> Open and • Enters the following path • "c:/../cme/Example/GCMLX.cml"(relative path) . • L1 clicks on "Open".
Expected Output	An error message is displayed "There is no editor registered for the file: c://cme/Example/GCMLX.cml.

Test case to convert the model from the GCML to XCML.

Test ID	TCS_3.1(Convert the model form the GCML to XCML) (Sunny Day 1)
Purpose	To test if a user can fetch the existing XCML from repository when the XCML already exists in the Repository.
Preconditions	The user(“L1”) is already logged on to the system. The GCML and its corresponding XCML already exist in the repository.
Input	• L1 clicks on Model-> GCML to XCML • .L1 clicks on Browse • Browses for “c:/../cme/Example/GCML1.gcml_diagram” present in the repository • L1 browses for “c:/../cme /Example/GCML1.gcml” • L1 browses for the destination path to store the XCML “c:/../cme /Example/XCML1.xcml • L1 clicks on “Save”. • The XCML already exists in the repository
Expected Output	“Transformation Complete”

Test ID	TCS_3.2(Convert the model form the GCML to XCML)(sunny day 2)
Purpose	To test if a user can convert the model from the GCML to XCML
Preconditions	The user "L1" is already logged on to the system. The XCML does not exist in the repository.
Input	• L1 clicks on Model-> GCML to XCML • .L1 clicks on Browse • Browses for "c:/../cme/Example/GCML1.gcml_diagram" present in the repository • L1 browses for "c:/../cme /Example/GCML1.gcml" • L1 browses for the destination path to store the XCML "c:/../cme /Example/XCML1.xcml L1 clicks on "Save".
Expected Output	The XCML file is created and saved on to the repository and popup "Transformation Complete".

Test ID	TCS_3.3(Convert the model form the GCML to XCML)
Purpose	To test if a L1 can convert the model from the GCML to XCML
Preconditions	• The user "L1" is already logged on to the system. The XCML does not exist in the repository.
Input	• L1 clicks on Model-> GCML to XCML • L1 clicks on Browse • Browses for "c:/../cme/Example/GCML1.gcml_diagram" present in the repository • L1 browses for "c:/../cme /Example/GCML1.gcml" • L1 Clicks on "Save"

Expected Output	Error “Please select the path to store the XCML file”

Test case for Edit Model Meta Data

Sunny Day: The user tries to edit the model metadata that is already present in the repository

Test ID	TCS_4.1(Edit model meta Data)
Purpose	To test if a user can edit the model meta data.
Preconditions	• The user L1 is already logged on to the system. • There are only a few meta data properties that can be edited by L1. • The remaining are set not to change (protected properties) for the given domain.
Input	• L1 browses for model (GCML) present in the repository. • L1 clicks on Model Administration • L1 selects “EditMetaData”. • L1 changes the ‘audio’ property of the Device object.
Expected Output	The model has a new audio property.

Sunny Day 2: The user tries to edit the model metadata that is already present in the repository

Test ID	TCS_4.2(Edit model meta Data)
Purpose	To test if a user can edit the model meta data.
Preconditions	The user(“L1”) is already logged on to the system. There are only a few meta data properties that can be edited by the user.

	The remaining are set not to change (protected) for the given domain.
Input	• L1 browses for model (GCML) present in the repository. • L1 clicks on Model Administration • L1 selects “EditMetaData”. • L1 changes the capability of the Device object.
Expected Output	The model has a new capability

Rainy Day: The user tries to edit the model metadata which is protected

Test ID	TCS_4.3(Edit model meta Data)
Purpose	To test the user should not be able to edit the model meta data that is protected.
Preconditions	The user(“L1”) is already logged on to the system. There are only a few meta data properties that can be edited by the user. The remaining are assumed not to change in the given domain.
Input	• L1 browses for model (GCML) present in the repository. • L1 clicks on Model Administration • L1 selects “EditMetaData”. • L1 tries to delete the path, that gives the file’s location
Expected Output	An error message is popped up, “ The path cannot be left blank, Please enter the appropriate location”

Test cases to convert from XCML to GCML

Sunny day 1: To convert the GCML model to XCML that is already present in the repository.

Test ID	TCS_5.1(Convert the model form the XCML to GCML)(sunny day 1)
Purpose	To test if a user can fetch the existing GCML from repository when the GCML already exists in the Repository.
Preconditions	The user("L1 ") is already logged on to the system. The XCML and its corresponding GCML file " GCML3.gcml_diagram" already exist in the repository.
Input	• The user "L1", click on File • "L1" clicks on "Open" • The user browses for the XCML form "c:/../cme/Example/xcml1.xcml • The user browses for the GCML diagram path "c:/../cme/Example/GCML/Gcml1.gcml_diagram • The user browses for the GCML "c:/../cme/Example/ GCML/Gcml1.gcml" • L1 selects on "convertXCMLToGCML".
Expected Output	The GCML file GCML1.gcml and the diagram GCML1.gcml_diagram is fetched from the repository Note: Since the XCML file from Team 1 has has fewer attributes than required when compared with the GCML's requirement, we get some errors during the conversion of Team1's XCML to GCML

Test ID	TCS_5.2(Convert the model form the XCML to GCML)(sunny day 2)
Purpose	To test if a user can convert the model from the XCML to GCML
Preconditions	The user(“L1”) is already logged on to the system. The GCML does not exist in the repository.
Input	• The user “L1”, click on File • “L1” clicks on “Open” • The user browses for the XCML form “c:/../cme/Example/xcml1.xcml • The user browses for the GCML diagram path “c:/../cme/Example/GCML/Gcml1.gcml_diagram • The user browses for the GCML “c:/../cme/Example/GCML/Gcml1.gcml” • L1 selects on “convertXCMLToGCML”. • The file is converted
Expected Output	The GCML file GCML1.gcml and the diagram file GCML1.gcml_diagram is stored in the location chosen by L1 Note: Since the XCML file from Team 1 has has fewer attributes than required when compared with the GCML’s requirement, we get some errors during the conversion of Team1’s XCML to GCML

.

Test ID	TCS_5.3(Convert the model form the XCML to GCML)(rainy day)
Purpose	To test if a user can convert the model from the XCML to GCML
Preconditions	The user("L1") is already logged on to the system. The GCML does not exist in the repository.
Input	• The user "L1", click on File • "L1" clicks on "Open" • The user browses for the XCML form "c:/../cme/Example/xcml1.xcml • The user browses for the GCML diagram path "c:/../cme/Example/GCML/Gcml1.gcml_diagram • L1 selects on "convertXCMLToGCML". • The file is converted
Expected Output	Error "Please Select the path to store the GCML diagram"Note: Since the XCML file from Team 1 has has fewer attributes than required when compared with the GCML's requirement, we get some errors during the conversion of Team1's XCML to GCML

Test case to validate a model

Test ID	TCS_6.1(ValidateModel)
Purpose	To validate a GCML model present on canvas
Preconditions	• User L1 is already logged into the system with the valid user name and password • L1 clicks on the File • L1 selects Open • L1 browses for the GCML file • Opens an already existing GCML file GCML1.gcml_diagram
Input	Complete Model GCML1.gcml_diagram is opened on the Canvas
Expected Output	No node has a red cross on it

Test ID	TCS_6.2 (ValidateModel)
Purpose	To validate a GCML model which is Created
Preconditions	User L1 is logged into the system with the valid userid and password. L1 drags the person box on to the canvas, enters the person id = Lazaro, person name = “Lazaro”, person role = “user” L1 drags Isattached on to the canvas and links person to Is attached with PersonToIsAttached link. L1 drags device box to the canvas, and enters Device Capability =”Audio”, IsLocal =”True “ IsVirtual= ”True” L1 drags the link IsAttachedToDevice to attach the device and IsAttached L1 drags Connection on to the canvas, connects the Device and the Connection with the DeviceToConnection link L1 repeats the steps 1 to 5 for person 2 with a different person credentials.{name=Leandro, role=user,id = leandro},Device Capability = audio, Islocal=false, Isvirtual=true. L1 clicks on Diagram - >selects validate.
Input	Model is created and is on the ME
Expected Output	No red cross are seen on the model

Test ID	TCS_6.3 (ValidateModel)
Purpose	To validate a GCML model present on canvas
Preconditions	User L1 is logged into the system with the valid userid and password. L1 drags the person box on to the canvas, enters the person id = Lazaro, person name = “”, person role = “user” L1 drags Isattached on to the canvas and links person to Is attached with PersonToIsAttached link. L1 drags device box to the canvas, and enters Device Capability =”Audio”, IsLocal =”True “ IsVirtual= ”True” L1 drags Connection on to the canvas, connects the Device and the Connection with the DeviceToConnection link L1 repeats the steps 1 to 5 for person 2 with a different person credentials
Input	Incomplete Complete Model drawn on the canvas
Expected Output	Red cross is seen on Person with PersonId = “Lazaro”

Test cases to add model to repository.

Test ID	TCS_7.1 (Add model to repository)(sunny day 1)
Purpose	To test the addition of model to Repository
Preconditions	• L1 has already logged into the system. • L1 draws a model using steps in preconditions of TCS_6.1.
Input	• L1 clicks on Model -> save to repository • L1 enters the GCML5.gcml as the file name
Expected Output	GCML5.gcml is saved in repository. Message "Model Saved"

Test ID	TCS_7.2 (Add Model to repository)(sunny day 2)
Purpose	To test addition of a model with pre-existing name in Repository
Preconditions	• L1 has already logged into the system. • L1 draws a model using steps in preconditions of TCS_6.1. • Repository contains a model named GCML5.gcml
Input	• L1 clicks on Model -> save to repository • L1 enters the GCML5.gcml as the file name • L1 clicks yes when prompted.
Expected Output	DialogMsg ' Do you want to overwrite'. The new model is saved.

Test ID	TCS_7.3 (Add model to repository)(rainy day)
Purpose	Adding incomplete model to the repository.
Preconditions	The user 'L1' is already logged into the system. • L1 clicks file ->open and browses for " The user browses for the GCML file /../cme/example/GCML_incomplete.gcml_diagram
Input	• L1 clicks on Model - >Save model to repository.
Expected Output	• The system displays a message: "Invalid file". • The model is not saved in the repository.

7.2. Sub-System Tests.

Subsystem test cases for package cme.uci

Test ID	SST_1.1(Convert the model form the GCML to XCML)
Purpose	To test the transformation package correctly carries out the functionality
Preconditions	None.
Input	• Run Transform.Java(String args[]): o Args[0]= "UseCorrectFolder" ; o Args[1]= "-option(fromGcmltoXcml)" ; o Args[2] = "GcmlSample.gcml";
Expected Output	There is a file out.xcml with the correct transformed model.

	This is verified by using it as the input for the next test case.

Test ID	SST_1.2(Convert the model form the XCML to GCML)

Purpose	To test the transformation package correctly carries out the functionality
Preconditions	None.
Input	• Run Transform.Java(String args[]): o Args[0]= "UseCorrectFolder" ; o Args[1]= "-option(fromXcmltoGcml)" ; o Args[2] = "xcmlSample.xcml";
Expected Output	There is a file out.gcml with the correct transformed model. This is verified by using it as the input for the next test case.

7.3. Unit Tests.

The unit test cases have been tested using junit. These test cases are method of testing that verifies the individual units of source code are working properly. We have run the test and all have passed and the results screen shot is attached below.

Test ID	UT_1.1

Code	
	```
	/* (non-Javadoc)
		 * @see junit.framework.test_01
		 * The xcml contains one schema, the schema contains three people
		 *
		 */
		public void test_01() throws Exception
		{
			int expected = 3;
			assertEquals( expected,
					xcml.getUserSchema().
					getPerson().size());
		}
	```

Result	• Pass

Test ID	UT_1.2

Code	`/* (non-Javadoc)`
	`        * @see junit.framework.test_02`
	`        * the xcml contains one schema, the schema contains three devices`
	`        */`
	`       public void test_02() throws Exception`
	`       {`
	`              int expected = 3;`
	
	`              assertEquals( expected,`
	`                                    xcml.getUserSchema().`
	`                                    getConnection().get(0).`
	`                                    getDevice().size());`
	`       }`

Result	• Pass

Test ID	UT_1.3

Code	`/* (non-Javadoc)`
	`        * @see junit.framework.test_03`
	`        * the xcml contains one schema, the schema contains the first person`
	`        * has person name = Dr. Wong`
	`        * personId = 023`
	`        */`
	`       public void test_03() throws Exception`
	`       {`
	`              String expectedName = "Dr. Wong";`
	`              String expectedId   = "023";`
	
	`              assertEquals( expectedName,`
	`                                    xcml.getUserSchema().`
	`                                    getPerson().get(0).`
	`                                    getPersonName());`
	`              assertEquals( expectedId,`
	`                                    xcml.getUserSchema().`
	`                                    getPerson().get(0).`
	`                                    getPersonID());`
	`       }`

Result	• Pass

Test ID	UT_1.4

Code	`/* the xcml contains one schema, the schema contains the first person`
	`        * has person name = Dr. Wong`
	`        * personId = 023`
	`        */`
	`      public void test_04() throws Exception`
	`      {`
	`            String expectedName = "Dr. Pi";`
	`            String expectedId   = "041";`
	
	`            assertEquals( expectedName,`
	`                              xcml.getUserSchema().`
	`                              getPerson().get(1).`
	`                              getPersonName());`
	`            assertEquals( expectedId,`
	`                              xcml.getUserSchema().`
	`                              getPerson().get(1).`
	`                              getPersonID());`
	`      }`

Result	• Pass

Test ID	UT_1.5

Code	`/*The xcml contains one connection, with ID = "connection1"`
	`        *`
	`        */`
	`      public void test_05() throws Exception`
	`      {`
	`            String expectedId = "connection1";`
	
	`            assertEquals( expectedId,`
	`                              xcml.getUserSchema().`
	`                              getConnection().`
	`                              get(0).getConnectionID());`
	`      }`

Result	• Pass

Test ID	UT_1.6

Code	`/*The xcml contains one medium of type = "LiveAV"`
	`*`
	`*/`
	`public void test_06() throws Exception`
	`{`
	`String expectedId = "LiveAV";`
	
	`assertEquals( expectedId,`
	`xcml.getUserSchema().`
	`getMediumType().get(0).`
	`getMediumTypeName());`
	`}`

Result	• Pass

Test ID	UT_1.7

Code	`/*The xcml contains one medium of type = "Generic"`
	`*`
	`*/`
	`public void test_07() throws Exception`
	`{`
	`String expectedId = "Generic";`
	
	`assertEquals( expectedId,`
	`xcml.getUserSchema().`
	`getFormType().get(0).`
	`getFormTypeName());`
	`}`

Result	• Pass

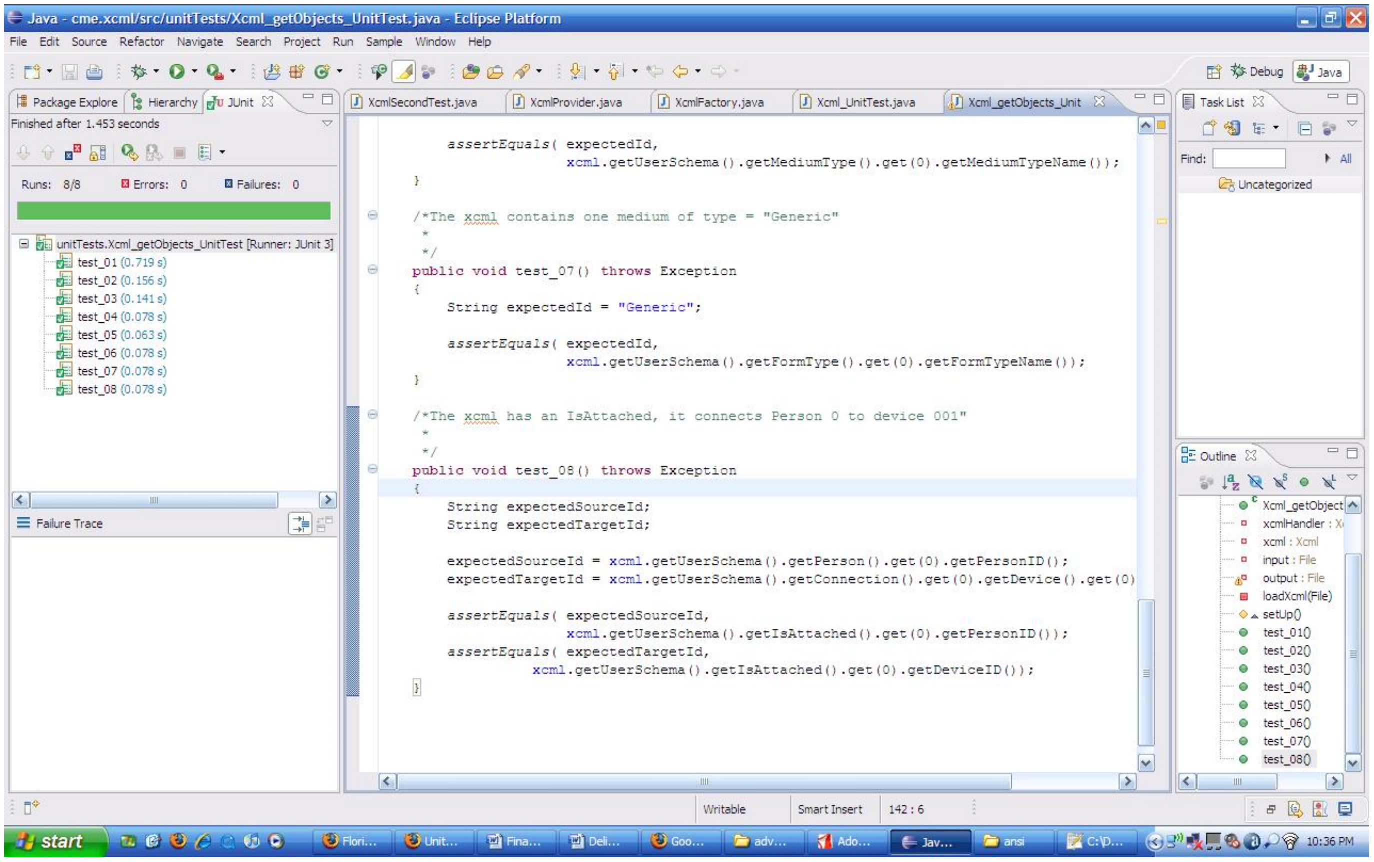
Test ID	UT_1.8

Code

```
/*The xcml has an IsAttached, it connects Person 0 to
device 001"
       *
       */
      public void test_08() throws Exception
      {
            String expectedSourceId;
            String expectedTargetId;


            expectedSourceId = xcml.getUserSchema().
                        getPerson().get(0).
                        getPersonID();

            expectedTargetId = xcml.getUserSchema().
                               getConnection().get(0).
                               getDevice().get(0).
                               getDeviceID();

            assertEquals( expectedSourceId,
                                 xcml.getUserSchema().
                                 getIsAttached().get(0).
                                 getPersonID());
            assertEquals( expectedTargetId,
                          xcml.getUserSchema().
                          getIsAttached().get(0).
                          getDeviceID());
      }
```

Result	• Pass

Screen Shot of Unit Test Cases:



```
Finished after 1.453 seconds

Runs: 8/8    Errors: 0    Failures: 0

unitTests.Xcml_getObjects_UnitTest [Runner: JUnit 3]
    test_01 (0.719 s)
    test_02 (0.156 s)
    test_03 (0.141 s)
    test_04 (0.078 s)
    test_05 (0.063 s)
    test_06 (0.078 s)
    test_07 (0.078 s)
    test_08 (0.078 s)
```

```
        assertEquals( expectedId,
                      xcml.getUserSchema().getMediumType().get(0).getMediumTypeName());
    }

    /*The xcml contains one medium of type = "Generic"
     *
     */
    public void test_07() throws Exception
    {
        String expectedId = "Generic";

        assertEquals( expectedId,
                      xcml.getUserSchema().getFormType().get(0).getFormTypeName());
    }

    /*The xcml has an IsAttached, it connects Person 0 to device 001"
     *
     */
    public void test_08() throws Exception
    {
        String expectedSourceId;
        String expectedTargetId;

        expectedSourceId = xcml.getUserSchema().getPerson().get(0).getPersonID();
        expectedTargetId = xcml.getUserSchema().getConnection().get(0).getDevice().get(0)

        assertEquals( expectedSourceId,
                      xcml.getUserSchema().getIsAttached().get(0).getPersonID());
        assertEquals( expectedTargetId,
                  xcml.getUserSchema().getIsAttached().get(0).getDeviceID());
    }
```

```
Xcml_getObject
    xcmlHandler : X
    xcml : Xcml
    input : File
    output : File
    loadXcml(File)
    setUp()
    test_01()
    test_02()
    test_03()
    test_04()
    test_05()
    test_06()
    test_07()
    test_08()
```

7.4. Evaluation of Tests

Unique ID	Test Result	Actual Result
TCS_1.1	Pass	CME is launched
TCS_1.2	Pass	CME is launched
TCS_1.3	Pass	Invalid UserName or Password
TCS_2.1	Pass	GCML1.gcml_diagram is displayed on CME
TCS_2.2	Pass	GCML1.gcml_diagram and GCML2.gcml_diagram are displayed on CME
TCS_2.3	Pass	Error Message " There is no editor Registered for the file GCMLx.cml"
TCS_3.1	Pass	Transformation Complete
TCS_3.2	Pass	Transformation Complete
TCS_3.3	Pass	Cannot find the XCML path
TCS_4.1	Fail	Not Implemented yet
TCS_4.2	Fail	Not Implemented yet
TCS_4.3	Fail	Not Implemented yet
TCS_5.1	Pass	Transformation Complete Note: We need to make changes to our GCML to matched Team's 1 XCML schema.
TCS_5.2	Pass	Transformation Complete Note: We need to make changes to our GCML to matched Team's 1 XCML schema.
TCS_5.3	Pass	Cannot find the GCML path Note: We need to make changes to our GCML to matched Team's 1 XCML schema.
TCS_6.1	Pass	No node has a red cross on it
TCS_6.2	Pass	No node has a red cross on it

TCS_6.3	Pass	Red cross is seen on Person with PersonId = “Lazaro”
TCS_7.1	Pass	GCML5.gcml is saved.
TCS_7.2	Pass	DialogMsg ' Do you want to overwrite'. The new model is saved.
TCS_7.3	Pass	Message “Invalid File”
SST_1.1	Pass	File out.xcml was created
SST_1.2	Pass	File out.gcml was created
UT_1.1 to UT_1.8	Pass	Screen shots provided.

8. Glossary

Person: Participant involved in a communication.

Device: Represents the virtual device used by a person during communication, e.g., cell phone, PDA, or computer. Devices have different capabilities to support different built-in medium types.

Connection: a channel in which the information exchange will occur. Constraints such as security, quality of service and other requirements can be applied to the connection.

Medium: A data piece or data stream, like file (e.g. Text file, Binary file), text, or a live stream (live audio).

Form: A container for other forms and / or media to be shared during a communication event.

Model Transformer: This component receives a model and translates it to the corresponding functionality, usually as calls to the Communication Engine.

EMF: Eclipse Modeling Framework is an Eclipse-based modeling framework and code generation facility for building tools and other applications based on a structured data model.

UML: UML (Unified Modeling Language) is a language used to visualize, specify, construct, and document the artifacts of a software-intensive system.

CML: Communication Modeling Language, a DSL to specify the Communication domain.

CVM: Communication Virtual Machine

GCML: Graphical CML. Communication Modeling Language represented as Graphs.

GEF: Graphical Editing Framework

GMF: Graphical Modeling Framework

NCB: Network Communication Broker

SE: Synthesis Engine

UCI: User Communication Interface.

CME: Communication Modeling Environment. The system described in this document.

XML: Extensible Markup Language

XCML: Communication Modeling Language represented in XML

9.Signatures

DEVELOPER: TEAM 2

Print Name: Lazaro Pi	Authorized Signature :
Print Name: Leandro Rafael Wong	Authorized Signature :
Print Name: Sandeep Varry	Authorized Signature :
Print Name: Manasa Bharadwaj	Authorized Signature :
Print Name: Jorge Jauregui	Authorized Signature :
Print Name: Roberto Espinoza	Authorized Signature :
Print Name: Marc Gauthier	Authorized Signature :

10. Appendix

10.1. Appendix A – Project Schedule

Task	Milestone	Task Name	Start	Finish	Predec.	Duration
1	No	Analysis/Software Requirements	9/3/2008	10/6/2008		24 days
2	No	Generate Use Cases	9/9/2008	9/15/2008		5 days
3	No	Primary Project Schedule	9/3/2008	9/8/2008		4 days
4	No	Identify Hardware, Software Resources	9/16/2008	9/22/2008	2	5 days
5	Yes	Review Use Cases	9/16/2008	9/16/2008	2	1 day
6	No	Identify milestones	9/16/2008	9/17/2008	2,5	1.5 days
7	No	Create Cost Analysis	9/17/2008	9/22/2008	5	4 days
8	No	Obtain approvals to proceed	9/22/2008	9/22/2008	7	1 day
9	No	Create Object and Dynamic diagrams	9/16/2008	9/18/2008	5	2.5 days
10	Yes	Create SRD deliverable	10/1/2008	10/2/2008	5	1.5 days
11	No	Create Presentation	10/6/2008	10/6/2008	10	1 day
12	No	Design	10/7/2008	11/3/2008		20 days
13	No	Review software specifications	10/7/2008	10/10/2008	11	3.5 days
14	No	Create Architectural Design	10/9/2008	10/15/2008	13	4 days
15	No	Create Object Design	10/9/2008	10/23/2008	13	10 days
16	Yes	Review Design with Team	10/23/2008	10/28/2008	15,14	3 days
17	No	Obtain approval to proceed	10/28/2008	10/28/2008	16	0.5 days
18	Yes	Create DD deliverable	11/3/2008	11/3/2008	17	0 days
19	No	Implementation	10/28/2008	11/27/2008		22.5 days
20	No	Review System / Object Design	10/28/2008	10/31/2008	16	3.5 days
21	Yes	Implement Import module	10/28/2008	11/25/2008	16	20 days
22	Yes	Implement Convert	10/28/2008	11/11/2008	16	10 days
23	No	Unit Testing	10/31/2008	11/6/2008	16FS+3	4 days
24	No	Functional Testing	11/4/2008	11/27/2008	21FS-75%	17 days
25	No	Development complete	11/27/2008	11/27/2008	24	0.5 days
26	No	Final Deliverable	11/4/2008	12/1/2008		20 days
27	No	Develop Final Software documentation	11/4/2008	11/10/2008	12	5 days
28	No	Review of Final Deliverable	11/11/2008	11/11/2008	27	0.5 days
29	No	Final Presentation	11/28/2008	12/1/2008	19	2 days

10.2. Appendix B – Use Cases

Explanations:

Actors: Modeler and Administrator. For misuse cases: Hacker

Use Case ID: The use case ID has been coded as follows:

Team + "_" + SystemName +"_" + UseCaseLevel + "_" + IncrementalNumericID + "_" + UseCaseName

Example: T2_CME_SUB_01_CreateModel

This can be interpreted as a use case by Team 2 for the Communication Modeling Environment system. This is a functional sub-use case, has numeric id 01, and relates to the creation of a model.

Use Case Level Abbreviations:

HIGH = High Level Use Case

SYSE2E = System Level End to End Use Case

SUB = Functional Sub-use Case

10.2.1.1. Drag Object to Canvas

Use Case ID: T2_CME_SUB_01_DragObjectToCanvas
Use Case Level: Functional Sub-use Case

Details

Actors: Modeler

Pre-conditions: Modeler has the Communication Modeling Environment (CME) open and he is editing a Model.

Description: The Modeler will drag and drop shape to the Canvas.

Trigger: The Modeler visually selects the shape he wants to add to the model.

The system responds by …

1. Modeler clicks on the toolbox item representing the shape and without releasing the mouse drags the cursor over the canvas.
2. The CME validates that the shape can be added to the canvas in the area selected by Modeler.
3. The CME draws the shape on the canvas.
4. Modeler saves the Model (click save button on Application menu)

Post-conditions: the model contains the newly added shape.

Alternative courses of action: None

Extensions: None

Exceptions:

- The Modeler can drop the shape in an illegal area; in this case the CME will issue an error.

Related use cases:

- T2_CME_SUB_02_CreateEdgeBetweenObjects

Decision support

Frequency: Moderate to High - (average of 10 shapes per model) Performed for all shapes to be added to a model, but the creation of models is infrequent (10 models per day).

Criticality: High - Directly support creation of models (one of the main uses of the system).

Risk: Medium

Constraints:

Usability

- No previous training is needed.

- Drag and drop functionality will be provided by the Communication Modeling Environment.

Reliability

- Mean Time to Failure – 1% failure for every 24 hours of operation is acceptable.

Performance

- The created model shall be saved within 2 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 09/18/2008

Date Last Modified: 09/21/2008

10.2.1.2. Create Edge between Objects

Use Case ID: T2_CME_SUB_02_CreateEdgeBetweenObjects

Use Case Level: Functional Sub-use Case

Details

Actors: Modeler

Pre-conditions: Modeler has the Communication Modeling Environment (CME) open and he is editing a Model.

Description: The Modeler will connect two shapes on the Canvas by creating a line from one to the other.

Trigger: The Modeler visually selects the shapes he wants to connect and decides the direction of the connection.

The system responds by …

1. Modeler clicks on the start shape and without releasing the mouse drags the cursor to the end shape and then releases the click.
2. The CME validates that the shapes can be connected according to the metamodel (it also checks the direction).
3. The CME draws the edge between the 2 shapes.
4. Modeler saves the Model (click save button on eclipse menu)

Post-conditions: the model contains the newly added edge (connection).

Alternative courses of action: None

Extensions: None

Exceptions:

- The Modeler can try to connect 2 shapes against the metamodel definition; in this case the CME will issue an error.

Related use cases:

- T2_CME_SUB_01_DragObjectToCanvas

Decision support

Frequency: Moderate to High - (average of 10 edges per model) Performed for all edges to be added to a model, but the creation of models is infrequent (10 Models per day).

Criticality: High - Directly support creation of models (one of the main uses of the system).

Risk: Medium

Constraints:

Usability

- No previous training is needed.
- Drag and drop functionality will be provided by the Communication Modeling Environment.

Reliability

- Mean Time to Failure – 1% failure for every 24 hours of operation is acceptable.

Performance

- The created model shall be saved within 2 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 09/18/2008

Date Last Modified: 09/21/2008

10.2.1.3. Create New Empty Model

Use Case ID: T2_CME_HIGH_03_CreateNewEmptyModel

Use Case Level: High Level Use Case

Details

Actors: Modeler

Pre-conditions: Modeler has the Communication Modeling Environment (CME) open.

Description: The Modeler will create a new model.

Trigger: The Modeler clicks the "New" menu item in the "File" menu of the CME.

The system responds by …

1. CME presents the Create New Diagram Dialog Wizard to the user (select file that will contain Diagram Model page).
2. In the "File" text box, the Modeler enters the fully qualified name of the Layout file that will contain layout information for the new model.
3. The Modeler clicks "Next" button.
4. CME presents the Create New Diagram Dialog Wizard to the user (select file that will contain Domain Model page).

5. In the “File” text box, the Modeler enters the fully qualified name of the object data file that will contain object information for the new model.
6. The Modeler clicks “Finish” button.
7. The system presents a dialog box saying Creating Diagram and Model files while those files are being created.
8. The CME system opens the newly created empty file in the editor.

Post-conditions: the two files representing the model are saved in the file system and the Model is open in the editor ready to be edited. The model is empty (no shapes or edges).

Alternative courses of action:

At steps 3 to 6 the Modeler can hit “Back” button to go to the previous step in the Wizard.

At steps 2 to 6 the Modeler can hit “Cancel” button to cancel the creation of the new model.

At steps 2 and 5 the Modeler can hit the “Browse” button to visually pick the location of the new files.

Extensions:

- T2_CME_HIGH_04_CreateGenericModel

Exceptions:

- The Modeler can enter a name that already exists. The CME will warn the modeler that this action will overwrite the exiting model.

Related use cases:

- T2_CME_SUB_01_DragObjectToCanvas
- T2_CME_SUB_02_CreateEdgeBetweenObjects

Decision support

Frequency: Moderate - Performed for all newly created models (10 Models per day).

Criticality: High - Directly support creation of models (one of the main uses of the system).

Risk: Medium

Constraints:

Usability

- No previous training is needed.
- Drag and drop functionality will be provided by the Communication Modeling Environment.

Reliability

- Mean Time to Failure – 1% failure for every 24 hours of operation is acceptable.

Performance

- The created model shall be saved within 2 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 09/18/2008

Date Last Modified: 09/21/2008

10.2.1.4. Create Generic Model

Use Case ID: T2_CME_HIGH_04_CreateGenericModel

Use Case Level: High Level Use Case (abstract)

Details

Actors: Modeler

Pre-conditions: Modeler has the Communication Modeling Environment (CME) open.

Description: The Modeler will create and save a new model and add shapes, edges, and will input desired values.

Trigger: The Modeler clicks the "New" menu item in the "File" menu of the CME.

The system responds by …

1. All steps are necessary to have newly created empty file in the editor (as specified in use case T2_CME_HIGH_03_CreateNewEmptyModel).

2. Modeler decides what he needs to model and identifies shapes, edges, and values.
3. For all shapes, modeler adds the shape to the canvas (T2_CME_SUB_01_DragObjectToCanvas).
4. For all edges, modeler connects source shape to target shape with the selected edge (T2_CME_SUB_02_CreateEdgeBetweenObjects).
5. Modeler sets all relevant properties with desired values.
6. Modeler saves the completed model by clicking the save button.

Post-conditions: the two files representing the model are saved in the file system and the Model is open in the editor ready to be edited. The model is contains all shapes, edges and values.

Alternative courses of action:

None specific to this level.

Extensions:

- Concreted by T2_CME_SYSE2E_05_CreateGroupChat
- Concreted by T2_CME_SYSE2E_06_Create2WayVoiceConnection
- Concreted by T2_CME_SYSE2E_07_CreateFileTransfer

Exceptions:

- None specific to this level.

Related use cases:

- Uses T2_CME_SUB_01_DragObjectToCanvas
- Uses T2_CME_SUB_02_CreateEdgeBetweenObjects

Decision support

Frequency: Moderate - Performed for all newly created models (10 Models per day).

Criticality: High - Directly support creation of models (one of the main uses of the system).

Risk: Medium

Constraints:

Usability

- No previous training is needed.
- Drag and drop functionality will be provided by the Communication Modeling Environment.

Reliability

- Mean Time to Failure – 1 failure for every 24 hours of operation is acceptable.

Performance

- The created model shall be saved within 2 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 09/18/2008

Date Last Modified: 09/21/2008

10.2.1.5. Open Existing GCML Model

Use Case ID: T2_CME_SYSE2E_05_OpenExistingModel

Use Case Level: System-Level End-to-End

Details

Actors: Modeler

Pre-conditions: Modeler has the Communication Modeling Environment (CME) open. There is an exciting model in GCML format and the modeler knows the full path to the model.

Description: The Modeler will open a previously saved model into the editor.

Trigger: The Modeler clicks the "Open" menu item in the "File" menu of the CME.

The system responds by …

1. CME presents the Open File Dialog.
2. The Modeler enters the fully qualified name of the previously saved Diagram file that he wants to open.
3. The Modeler clicks “Open” button.
4. The CME system opens the picked file in the editor.

Post-conditions: the Model is open in the editor and ready to be edited.

Alternative courses of action:

At steps 3 to 6 the Modeler can hit “Back” button to go to the previous step in the Wizard.

At steps 2 to 6 the Modeler can hit “Cancel” button to cancel the creation of the new model.

At steps 2 and 5 the Modeler can hit the “Browse” button to visually pick the location of the new files.

Extensions:

None.

Exceptions:

- The Modeler can enter a name that already exists. The CME will warn the modeler that this action will overwrite the exiting model.

Related use cases:

- T2_CME_SUB_01_DragObjectToCanvas
- T2_CME_SUB_02_CreateEdgeBetweenObjects

Decision support

Frequency: Moderate to High - Performed for every time the Modeler wants to open an existing model (80 times per day).

Criticality: High - Directly support edition of models (one of the main uses of the system).

Risk: Medium

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1% failure for every 24 hours of operation is acceptable.

Performance

- The created model shall be saved within 2 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 09/18/2008

Date Last Modified: 09/21/2008

10.2.1.6. Create Group Chat Model

Use Case ID: T2_CME_SYSE2E_06_CreateGroupChat

Use Case Level: System-Level End-to-End

Details

Actors: Modeler.

Pre-conditions: The Modeler has the Communication Modeling Environment (CME) Open and has access to save created models to the Repository.

Description: The Modeler wishes to create a 3 way chat model and he will drag and drop shapes to the Canvas in order to model three way chat communication.

Trigger: The Modeler Starts a new model by selecting the "New" option in the CME.

The system responds by …

1. The CME creates a new model as defined on T2_CME_HIGH_03_CreateNewModel.
2. The Modeler drags Participant to the canvas.
3. The Modeler drags isAttached to the canvas.

4. The Modeler creates edge from Person to isAttached.
5. The Modeler drags Device to the canvas.
6. The Modeler creates edge from isAttached_1 to Device_1.
7. Repeat steps 2 through 7 n times
8. The Modeler drags Medium to the canvas.
9. The Modeler drags Connection to the canvas.
10. The Modeler creates edges from Connection to Medium.
11. The Modeler creates edge from Connection to Device.
12. Repeat steps 11 n-times.
13. The Modeler saves the model to the repository.

Post-conditions: After this use case is complete, the created model has been saved to the repository and is ready to be used by the GUI user.

Alternative courses of action: The Modeler can instantiate all fields in the modeled objects (i.e.: set all properties) in which case the model created can be executed without prompting the final user to fill any other input.

Extensions: None

Exceptions: None

Related use cases:

- Extends T2_CME_HIGH_03_CreateNewModel
- T2_CME_SUB_01_DragObjectToCanvas
- T2_CME_SUB_02_CreateEdgeBetweenObjects

Decision Support

Frequency: Low - This model would have to be created only once, it will be modified sporadically.

Criticality: High - Directly support one of the main uses of the system

Risk: Medium

Constraints:

Usability

- No previous training is needed.
- Drag and drop functionality will be provided by the use of the Communication Modeling Environment.
- On the user interfaces, all the shapes must contain brief textual descriptions.

Reliability

- Mean Time to Failure – 1% failure for every 24 hours of operation is acceptable.

Performance

- The created model shall be saved within 2 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 09/17/2008

Date Last Modified: 09/21/2008

10.2.1.7. Create 2-Way Voice Connection Model

Use Case ID: T2_CME_SYSE2E_07_ Create2WayVoiceConnection

Use Case Level: *System-Level End-to-End*

Details

Actors: Modeler

Pre-conditions: Modeler has the Communication Modeling Environment (CME) open and has access to save created models to the model Repository ("<sysFolder>\CME\Repository\").

Description: The Modeler will drag and drop shapes to the Canvas in order to model two way voice communication.

Trigger: The Modeler Starts a new model by selecting the "New" option in the CME.

The system responds by …

1. The CME creates the new model as defined on T2_CME_HIGH_03_CreateNewModel.
2. Modeler drags Participant1 to the canvas and sets the Role property to "Caller".
3. Modeler drags Participant2 to the canvas and sets the Role Property to "Receiver".
4. Modeler Device1 to the canvas and set device capability to "LiveAudio".
5. Modeler Device2 to the canvas and set device capability to "LiveAudio".
6. Modeler drags Connection1 to the canvas and creates edges from Connection1 to Device1 and Device2.
7. Modeler drags isAttached1 to the canvas and creates edge from isAttached1 to Device1.
8. Modeler drags isAttached2 to the canvas and creates edge from isAttached2 to Device2.
9. Modeler saves the model to the repository.

Post-conditions: After this use case is complete, the created model has been saved to the repository and is ready to be used by the GUI user.

Alternative courses of action: The Modeler can instantiate all fields in the modeled objects (i.e.: set all properties) in which case the model created can be executed with prompting the final user to fill any other input.

Extensions: None

Exceptions: None

Related use cases:

- Extends T2_CME_HIGH_03_CreateNewModel
- T2_CME_SUB_01_DragObjectToCanvas
- T2_CME_SUB_02_CreateEdgeBetweenObjects

Decision Support

Frequency: Low - This model would have to be created only once, it will be modified sporadically.

Criticality: High - Directly support one of the main uses of the system

Risk: Medium

Constraints:

Usability

- No previous training is needed.
- Drag and drop functionality will be provided by the use of the Communication Modeling Environment.
- On the user interfaces, all the shapes must contain brief textual descriptions.

Reliability

- Mean Time to Failure – 1% failure for every 24 hours of operation is acceptable.

Performance

- The created model shall be saved within 2 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 09/17/2008

Date Last Modified: 09/21/2008

10.2.1.8. Create File Transfer Model

Use Case ID: T2_CME_SYSE2E_08_CreateFileTransfer

Use Case Level: System-Level End-to-End

Details

Actors: Modeler

Pre-conditions: Modeler has the Communication Modeling Environment (CME) Open and has access to save created models to the Repository.

Description: The Modeler drags Shapes and Edges to the CME in order to create a File Sharing Model that users will execute later.

Trigger: The Modeler Starts a new model by selecting the create new option in the CME, (view T2_CME_HIGH_03_CreateNewModel)

The system responds by …

1. The CME creates a new model as defined on T2_CME_HIGH_03_CreateNewModel.
2. The Modeler drags Person1 to the canvas.
3. The Modeler drags Person2 to the canvas.
4. The Modeler drags Device1 to the canvas and set device capability to TextFile.
5. The Modeler drags Device2 to the canvas and set device capability to TextFile.
6. The Modeler drags Medium1 to the canvas.
7. The Modeler drags Connection1 to the canvas and creates edges from Connection1 to Device1, Device2, and Medium1.
8. The Modeler drags isAttached1 to the canvas and creates edge from isAttached1 to Device1.
9. The Modeler drags isAttached2 to the canvas and creates edge from isAttached2 to Device2.
10. The Modeler saves the model to the repository.

Post-conditions: After this use case is complete, the created model has been saved to the repository and is ready to be used by the GUI user.

Alternative courses of action: The Modeler can instantiate all fields in the modeled objects (i.e. set all properties) in which case the model created can be executed with prompting the final user to fill any other input.

Extensions: None

Exceptions: None

Related use cases:

- Extends T2_CME_HIGH_03_CreateNewModel

- T2_CME_SUB_01_DragObjectToCanvas
- T2_CME_SUB_02_CreateEdgeBetweenObjects

Decision Support

Frequency: Low - This model would have to be created only once, it will be modified sporadically.

Criticality: High - Directly support one of the main uses of the system

Risk: Medium

Constraints:

Usability

- No previous training is needed.
- Drag and drop functionality will be provided by the use of the Communication Modeling Environment.
- On the user interfaces, all the shapes must contain brief textual descriptions.

Reliability

- Mean Time to Failure – 1% failure for every 24 hours of operation is acceptable.

Performance

- The created model shall be saved within 2 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 09/17/2008

Date Last Modified: 09/21/2008

10.2.1.9. Convert Model from GCML to XCML

Use Case ID: T2_CME_SYSE2E_09_ConvertFromGCMLtoXCML

Use Case Level: System-Level End-to-End

Details

Actors: Modeler

Pre-conditions: Modeler has the Communication Modeling Environment (CME) open and has a GCML file open in the CME.

Description: The modeler will give CME as input a GCML file and the CME will return the XCML version of the input file.

Trigger: The Modeler selects the Convert from GCML to XCML option on the menu.

The system responds by …

1. CME reads the GCML text into memory as a data structure of CML objects.
2. CME populates the XCML file with the CML objects following the CML v1.1 xml schema.
3. CME saves the XCML file in the repository.
4. CME returns the focus back to the canvas.

Post-conditions: the GCML version is still displayed on the CME canvas. The XCML file is saved in the repository.

Alternative courses of action:

None.

Extensions:

None.

Exceptions:

- None.

Related use cases:

- T2_CME_SYSE2E_10_ConvertFromGCMLtoXCML

Decision support

Frequency: Moderate to high - Performed for every time the Modeler wants to export an existing GCML model (12 times per day).

Criticality: Low. Only need to verify adherence to the CML v1.1 xml schema.

Risk: Low. It is mostly a straight forward approach.

Constraints:

Usability

- Internal to system.

Reliability

- Mean Time to Failure – 1 failure for every 120 hours of operation is acceptable.

Availability

- System should notify the user of a failure (incorrect format) gracefully without aborting the program

Performance

- Must take less than 15 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.
- Must be easily upgradeable to accommodate new shapes and objects.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Leandro Wong

Initiation Date: 10/06/2008

Date Last Modified: 10/06/2008

10.2.1.10. Convert Model from XCML to GCML

Use Case ID: T2_CME_SYSE2E_10_ConvertFromXCMLtoGCML

Use Case Level: System-Level End-to-End

Details

Actors: Modeler

Pre-conditions: Modeler has the Communication Modeling Environment (CME) open and has a previously validated (as explained in Validate Model) XCML file open in the XML editor of the CME (the XML text). Alternatively, the use case can be called from Import function and the XCML file can be selected in the file system.

Description: The modeler will give CME as input an XCML file and the CME will return the GCML version of the input file.

Trigger: The Modeler selects the Convert from XCML to GCML option on the menu.

The system responds by …

1. CME reads the XCML text into memory as a data structure of CML objects.
2. CME checks if it has a GCML file associated with this XCML file in the repository. If that is the case, CME compares the 2 files to see if they are equivalent (by comparing all objects).
3. If they are equivalent, CME uses the stored GCML and goes to step 8; otherwise it goes to step 4.
4. For each of the objects, CME calls the T2_CME_SUB_14_TransformObjectData to get the object data into GCML format.
5. For each of the objects, CME initializes an empty layout entry.
6. For each of the objects, CME will call the T2_CME_SUB_15_GenerateLayoutData to populate the layout data structure.
7. CME will use the object data and layout data to produce the GCML files.
8. CME opens the GCML file in the Modeling Environment.

Post-conditions: the GCML version of the file is generated and the model is displayed on the CME canvas.

Alternative courses of action:

This use case can be called from T2_CME_SYSE2E_12_ImportModelFromXCML; in this case, the use case returns the paths of the file to that use case so that it can be displayed as a graphical model.

Extensions:

None.

Exceptions:

- None.

Related use cases:

- Can be called by T2_CME_SYSE2E_12_ImportModelFromXCML
- T2_CME_SYSE2E_09_ConvertFromGCMLtoXCML

Decision support

Frequency: Moderate to high - Performed for every time the Modeler wants to import an existing XCML model (8 times per day).

Criticality: High. This is one of the 2 main uses of the system.

Risk: High. This requires algorithms and data structures that complicate the programming.

Constraints:

Usability

- Internal to system.

Reliability

- Mean Time to Failure – 1 failure for every 120 hours of operation is acceptable.

Availability

- System should notify the user of a failure (incorrect format) gracefully without aborting the program

Performance

- Must take less than 15 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.
- Must be easily upgradeable to accommodate new shapes and objects.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 10/06/2008

Date Last Modified: 10/06/2008

10.2.1.11. Add Model to Repository

Use Case ID: T2_CME_SYSE2E_11_AddModelToRepository

Use Case Level: System-Level End-to-End

Details

Actors: Modeler

Pre-conditions: Modeler has the Communication Modeling Environment (CME) open. There is an existing model in the file system and the modeler knows the full path to the model.

Description: The Modeler will add the existing model to the repository structure and will save additional metadata to describe the model.

Trigger: The Modeler clicks the "Models Administration" menu item in the CME.

The system responds by …

1. CME presents the Models Administration Window.
2. The Modeler clicks the Add Model Entry button.
3. The CME creates a new entry for the model and selects the entry in the window.
4. Modeler enters the name, description, fully qualified path and available formats into the appropriate text boxes.
5. The Modeler clicks "Save" button.
6. The CME system saves the entry into the repository data structure and saves a copy of the model into the internal repository folder.

Post-conditions: the Model is saved in the internal repository and there is an entry in the repository data structure to identify the Model.

Alternative courses of action:

At steps 2 to 5 the Modeler can hit "Cancel" button to cancel the creation of the new entry.

At step 4 the Modeler can hit the "Browse" button to visually pick the location of the model.

Extensions:

None.

Exceptions:

- The Modeler can enter a name that already exists. The CME will warn the modeler that this action will overwrite the exiting model.

Related use cases:

- T2_CME_SYSE2E_23_EditModelMetadata

Decision support

Frequency: Moderate - Performed for every time the Modeler wants to save a new model to the repository (10 times per day).

Criticality: High - Directly support one of the main uses of the system.

Risk: Medium

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1 failure for every 24 hours of operation is acceptable.

Performance

- The model shall be saved within 2 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 09/18/2008

Date Last Modified: 10/04/2008

10.2.1.12. Import Model from XCML

Use Case ID: T2_CME_SYSE2E_12_ImportModelFromXCML

Use Case Level: System-Level End-to-End

Details

Actors: Modeler

Pre-conditions: Modeler has the Communication Modeling Environment (CME) open.

Description: The Modeler will import an XCML file into the CME by picking the file on the File System.

Trigger: The Modeler clicks the "Import from XCML" menu item in the "File" menu of the CME.

The system responds by …

1. CME presents the Open File Dialog.
2. The Modeler enters the fully qualified name of the previously saved Diagram file that he wants to import.
3. The Modeler clicks "Open" button.
4. The XCML file is converted to GCML as explained in related use case T2_CME_SYSE2E_10_ConvertFromXCMLtoGCML .
5. The CME saves the files produced by the conversion into the repository.
6. The CME system opens the converted file in the editor.

Post-conditions: the Model is open in the editor and ready to be edited.

Alternative courses of action:

At step 2 the user can visually browse the file system to select the file to be imported.

In step 3 the Modeler has the option to cancel the import request.

Extensions:

None.

Exceptions:

- If the file name to be imported already exists (GCML, XCML or both), the user will be prompted to enter a different file name.
- The system could not load the XCML because it is in the wrong format.

Related use cases:

- T2_CME_SYSE2E_09_ConvertFromGCMLtoXCML
- T2_CME_SYSE2E_13_ImportModelFromGCML

Decision support

Frequency: Moderate - Performed for every time the Modeler wants to import an existing XCML model (10 times per day).

Criticality: High - Directly support import of models (one of the main uses of the system).

Risk: High. Implementing this use case is increased in difficulty due to the fact that the model should be rendered in as an aesthetically pleasing way as possible

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1 failure for every 24 hours of operation is acceptable.

Availability

- System should notify the user of a failure (incorrect format) gracefully without aborting the program

Performance

- The model shall be imported within 5 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Jorge Jauregui

Initiation Date: 09/18/2008

Date Last Modified: 09/21/2008

10.2.1.13. Import Model from GCML

Use Case ID: T2_CME_SYSE2E_13_ImportModelFromGCML

Use Case Level: System-Level End-to-End

Details

Actors: Modeler

Pre-conditions: Modeler has the Communication Modeling Environment (CME) open.

Description: The Modeler will import a GCML file into the CME by picking the file on the File System.

Trigger: The Modeler clicks the "Import from GCML" menu item in the "File" menu of the CME.

The system responds by …

1. CME presents the Open File Dialog.
2. The Modeler enters the fully qualified name of the previously saved Diagram file that he wants to import.
3. The Modeler clicks "Open" button.
4. The CME saves the files into the repository.
5. The CME system opens the file in the editor.

Post-conditions: the Model is open in the editor and ready to be edited.

Alternative courses of action:

At step 2 the user can visually browse the file system to select the file to be imported.

In step 3 the Modeler has the option to cancel the import request.

Extensions:

None.

Exceptions:

- If the file name to be imported already exists (GCML, XCML or both), the user will be prompted to enter a different file name.
- The system could not load the GCML because it is syntactically incorrect.

Related use cases:

- T2_CME_SYSE2E_09_ConvertFromGCMLtoXCML
- T2_CME_SYSE2E_12_ImportModelFromXCML

Decision support

Frequency: Moderate - Performed for every time the Modeler wants to import an existing GCML model (8 times per day).

Criticality: Low. The Modeler can already open file and then use save into a different location. It can also manually import in the file system and modify the repository files.

Risk: Low. The G-CML already contains the graphical representation of the communications model. Importing and rendering onto the canvas should be straightforward.

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1 failure for every 24 hours of operation is acceptable.

Availability

- System should notify the user of a failure (incorrect format) gracefully without aborting the program

Performance

- The model shall be imported within 5 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Jorge Jauregui

Initiation Date: 09/18/2008

Date Last Modified: 09/21/2008

10.2.1.14. Transform Object Data

Use Case ID: T2_CME_SUB_14_TransformObjectData

Use Case Level: Functional Sub-use Case

Details

Actors: Modeler

Pre-conditions: Modeler is executing T2_CME_SYSE2E_10_ConvertFromXCMLtoGCML.

Description: The object data information will be transformed from XCML to GCML format.

Trigger: The Modeler has started the T2_CME_SYSE2E_10_ConvertFromXCMLtoGCML use case. This use case calls T2_CME_SUB_14_TransformObjectData to get the object data in GCML format by passing the XCML format to it.

The system responds by …

1. Transform Object Data receives the object type and information.
2. Transform Object Data creates a new object based on the received information
3. The object is returned to the calling use case.

Post-conditions: the calling use case now has the object in the correct GCML format to match the corresponding object in the XCML file.

Alternative courses of action:

None

Extensions:

None.

Exceptions:

- None. Transformed Object Data is only called with valid communication object types.

Related use cases:

- Called by T2_CME_SYSE2E_10_ConvertFromXCMLtoGCML
- T2_CME_SYSE2E_12_ImportModelFromXCML

Decision support

Frequency: Moderate to high - Performed for every time the Modeler wants to import an existing XCML model (8 times per day). Average of 20 times per XCML imported (connections in CVM are objects too).

Criticality: High. Use case T2_CME_SYSE2E_10_ConvertFromXCMLtoGCML depends on this use case.

Risk: Medium. A data structure with mappings between the two formats needs to be developed, tested and maintained.

Constraints:

Usability

- Internal to system.

Reliability

- Mean Time to Failure – 1 failure for every 120 hours of operation is acceptable.

Availability

- System should notify the user of a failure (incorrect format) gracefully without aborting the program

Performance

- Really fast. Must take less than 0.45 second.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.
- Must be easily upgradeable to accommodate new shapes and objects.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Leandro Wong

Initiation Date: 10/05/2008

Date Last Modified: 10/05/2008

10.2.1.15. Generate Layout Data

Use Case ID: T2_CME_SUB_15_GenerateLayoutData

Use Case Level: Functional Sub-use Case

Details

Actors: Modeler

Pre-conditions: Modeler is executing T2_CME_SYSE2E_10_ConvertFromXCMLtoGCML.

Description: The layout data information will be generated in order to produce the complete GCML format version of the Model.

Trigger: The Modeler has started the T2_CME_SYSE2E_10_ConvertFromXCMLtoGCML use case. This use case calls T2_CME_SUB_15_GenerateLayoutData to get the layout data in GCML format by passing the current layout data structure (the layout info created so far) and the next object data to be added to the layout data structure.

The system responds by …

1. CME gets the shape and size of the object as explained in use case T2_CME_SUB_16_CalculateShapeAndSize.
2. CME generates the tentative coordinates according to the object's shape and size and the rest of the layout entries as explained in T2_CME_SUB_17_CalculateCoordinates.
3. CME check the complete layout for overlaps as explained in T2_CME_SUB_18_CheckOverlap.
4. CME returns the update list of layout entries with the added layout data to the calling use case T2_CME_SYSE2E_10_ConvertFromXCMLtoGCML.

Post-conditions: the calling use case now has the layout information for the current object as part of the layout data structure.

Alternative courses of action:

None.

Extensions:

None.

Exceptions:

- At step 3, the CME can report that there is overlap; in that case, the use case goes back to step 2 to generate a corrected set of coordinates and continue from there.

Related use cases:

- Called by T2_CME_SYSE2E_10_ConvertFromXCMLtoGCML
- T2_CME_SYSE2E_12_ImportModelFromXCML

Decision support

Frequency: Moderate to high - Performed for every time the Modeler wants to import an existing XCML model (8 times per day). Average of 10 times per XCML imported.

Criticality: High. Use case T2_CME_SYSE2E_10_ConvertFromXCMLtoGCML depends on this use case.

Risk: High, Generating layout is one of the most difficult parts of the project. A data structure with current layout data must be maintained.

Constraints:

Usability

- Internal to system.

Reliability

- Mean Time to Failure – 1 failure for every 120 hours of operation is acceptable.

Availability

- System should notify the user of a failure (incorrect format) gracefully without aborting the program

Performance

- Really fast. Must take less than 1 second.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.
- Must be easily upgradeable to accommodate new shapes and objects.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Leandro Wong

Initiation Date: 10/05/2008

Date Last Modified: 10/05/2008

10.2.1.16. Calculate Shape and Size

Use Case ID: T2_CME_SUB_16_CalculateShapeAndSize

Use Case Level: Functional Sub-use Case

Details

Actors: Modeler

Pre-conditions: Modeler is executing T2_CME_SUB_15_GenerateLayoutData.

Description: The shape and size of the object will be generated in order to produce the layout data information needed for the GCML format version of the Model.

Trigger: The Modeler has started the T2_CME_SUB_15_GenerateLayoutData use case. This use case calls T2_CME_SUB_16_CalculateShapeAndSize to get the shape and size of the current object.

The system responds by …

1. CME gets the default shape and size from the stored table of shape and sizes based on the object type.
2. CME adjusts up the Width and Height based on the current attribute values stored in the object instance (the length of any string values, e.g. the personName for the Person class).
3. CME adjust the size so that is within the Minimum and Maximum values as specified in the table of shapes and sizes.
4. CME returns the object layout data with the updated shape and size values.

Post-conditions: the calling use case now has the updated shape and size values for the current object as part of the layout data structure.

Alternative courses of action:

None.

Extensions:

None.

Exceptions:

- None.

Related use cases:

- Called by T2_CME_SUB_15_GenerateLayoutData
- Followed by T2_CME_SUB_17_CalculateCoordinates

Decision support

Frequency: Moderate to high - Performed for every time the Modeler wants to import an existing XCML model (8 times per day). Average of 10 times per XCML imported.

Criticality: High. Use case T2_CME_SYSE2E_10_ConvertFromXCMLtoGCML depends on this use case.

Risk: Medium, A data structure with default shape, size, minimum and maximum values for each of the object types must be developed, tested and maintained.

Constraints:

Usability

- Internal to system.

Reliability

- Mean Time to Failure – 1 failure for every 120 hours of operation is acceptable.

Availability

- System should notify the user of a failure (incorrect format) gracefully without aborting the program

Performance

- Really fast. Must take less than .25 second.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.
- Must be easily upgradeable to accommodate new shapes and objects.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Leandro Wong

Initiation Date: 10/05/2008

Date Last Modified: 10/05/2008

10.2.1.17. Calculate Coordinates

Use Case ID: T2_CME_SUB_17_CalculateCoordinates

Use Case Level: Functional Sub-use Case

Details

Actors: Modeler

Pre-conditions: Modeler is executing T2_CME_SUB_15_GenerateLayoutData.

Description: The coordinates of the object will be generated in order to produce the layout data information needed for the GCML format version of the Model.

Trigger: The Modeler has started the T2_CME_SUB_15_GenerateLayoutData use case. This use case calls T2_CME_SUB_16_CalculateCoordinates to get the coordinates of the current object.

The system responds by …

1. CME scans the layout data structure once to find the area where it can tentatively place the object (by going forward adding coordinates plus sizes until it gets to the current object).
2. CME uses the numbers found in step 1 along with the table of stored heuristics (a table of possible strategies to place the object based on its type and list of objects placed so far) to produce a set of possible coordinates.
3. CME returns the object layout data with the updated coordinates produced in step 2.

Post-conditions: the calling use case now has the coordinates for the current object as part of the layout data structure.

Alternative courses of action:

None.

Extensions:

None.

Exceptions:

- None.

Related use cases:

- Called by T2_CME_SUB_15_GenerateLayoutData
- Followed by T2_CME_SUB_18_CheckOverlap

Decision support

Frequency: Moderate to high - Performed for every time the Modeler wants to import an existing XCML model (8 times per day). Average of 10 times per XCML imported.

Criticality: High. Use case T2_CME_SYSE2E_10_ConvertFromXCMLtoGCML depends on this use case.

Risk: High. This requires a sophisticated algorithm and data structure with heuristics to be able to produce reasonable layout for potentially infinite many models.

Constraints:

Usability

- Internal to system.

Reliability

- Mean Time to Failure – 1 failure for every 120 hours of operation is acceptable.

Availability

- System should notify the user of a failure (incorrect format) gracefully without aborting the program

Performance

- Must take less than .50 second.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.
- Must be easily upgradeable to accommodate new shapes and objects.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 10/06/2008

Date Last Modified: 10/06/2008

10.2.1.18. Check Overlap

Use Case ID: T2_CME_SUB_18_CheckOverlap

Use Case Level: Functional Sub-use Case

Details

Actors: Modeler

Pre-conditions: Modeler is executing T2_CME_SUB_15_GenerateLayoutData. There is at least one object with complete layout data in the list of layout information.

Description: The CME will check that there is no overlap between the objects already placed (for which there is complete layout data).

Trigger: The Modeler has started the T2_CME_SUB_15_GenerateLayoutData use case. This use case calls T2_CME_SUB_18_CheckOverlap to check that there is no overlap in already placed shapes.

The system responds by …

1. CME scans the layout data structure once to find if there is any overlap (by going forward adding coordinates plus sizes until it gets to the current object).
2. CME returns the results of the scan done in step 1. (either NO OVERLAP or a list of the shapes that OVERLAP)

Post-conditions: the calling use case now has the coordinates for the current object as part of the layout data structure.

Alternative courses of action:

None.

Extensions:

None.

Exceptions:

- None.

Related use cases:

- Called by T2_CME_SUB_15_GenerateLayoutData
- Preceded by T2_CME_SUB_17_CalculateCoordinates

Decision support

Frequency: Moderate to high - Performed for every time the Modeler wants to import an existing XCML model (8 times per day). Average of 10 times per XCML imported.

Criticality: High. Use case T2_CME_SYSE2E_10_ConvertFromXCMLtoGCML depends on this use case.

Risk: Medium. This requires a somewhat complicated algorithm and data structure.

Constraints:

Usability

- Internal to system.

Reliability

- Mean Time to Failure – 1 failure for every 120 hours of operation is acceptable.

Availability

- System should notify the user of a failure (incorrect format) gracefully without aborting the program

Performance

- Really fast. Must take less than .25 second.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.
- Must be easily upgradeable to accommodate new shapes and objects.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Sandeep Varry

Initiation Date: 10/06/2008

Date Last Modified: 10/06/2008

10.2.1.19. Validate Model

Use Case ID: T2_CME_SYSE2E_19_ValidateModel

Use Case Level: System-Level End-to-End

Details

Actors: Modeler

Pre-conditions: Modeler has the Communication Modeling Environment (CME) open and has an XCML file open in the XML editor of the CME (the XML text). Alternatively, the use case can be called from “Validate Model from XCML File” and the file can be selected in the file system.

Description: The CME will check that the current model is valid.

Trigger: The Modeler clicks in “Validate Model” button.

The system responds by …

1. CME checks the model schema as defined in T2_CME_SUB_20_CheckModelSchema use case.

2. CME checks the semantic rules as defined in T2_CME_SUB_21_CheckSemanticRules.
3. The CME compiles both step 1 and 2 results and presents a report to the Modeler (either VALID or INVALID and list of the errors and warnings).

Post-conditions: the Modeler receives a report of the result of validation.

Alternative courses of action:

None.

Extensions:

None.

Exceptions:

- The same exceptions thrown by the sub use cases.

Related use cases:

- Uses T2_CME_SUB_20_CheckModelSchema.
- Uses T2_CME_SUB_21_CheckSemanticRules.

Decision support

Frequency: Moderate - Performed for every time the Modeler wants to validate an XCML model (10 times per day).

Criticality: High. The end product of CME must be a valid XCML model to be executed in the CVM so validation is very important.

Risk: High. Dependent in correct implementation of the 2 sub use cases.

Constraints:

Usability

- No previous training needed.

Reliability

- Mean Time to Failure – 1 failure for every 60 hours of operation is acceptable.

Availability

- System should notify the user of a failure gracefully without aborting the program

Performance

- Really fast. Must take less than .50 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

- The set of rules is read at runtime so that new rules can be added to the repository and the rest of the code does not have to change.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 10/06/2008

Date Last Modified: 10/08/2008

10.2.1.20. Check Model Schema

Use Case ID: T2_CME_SUB_20_CheckModelSchema

Use Case Level: Functional Sub-use Case

Details

Actors: Modeler

Pre-conditions: Modeler is executing T2_CME_SYSE2E_19_ValidateModel for a model in XCML format. The XSD schema for XCML has been previously developed and saved in the repository.

Description: The CME will check that the current model complies with XCML schema.

Trigger: The Modeler has started the T2_CME_SYSE2E_19_ValidateModel use case. This use case calls T2_CME_SUB_20_CheckModelSchema to check that the model complies with XCML schema.

The system responds by …

1. CME reads the saved XSD schema definition for CML that is saved in the repository.
2. CME uses the schema to validate the CML model.
3. CME returns the results of the validation done in step 2. (Either VALID or INVALID with the list of errors).

Post-conditions: the calling use case now has the results of the schema validation.

Alternative courses of action:

None.

Extensions:

None.

Exceptions:

- The schema is not found in the repository; in this case, CME will issue an appropriate error message and raise an exception back to the calling use case.

Related use cases:

- Called by T2_CME_SYSE2E_19_ValidateModel
- Followed by T2_CME_SUB_21_CheckSemanticRules

Decision support

Frequency: Moderate - Performed for every time the Modeler wants to validate an XCML model (10 times per day).

Criticality: High. The end product of CME must be a valid XCML model to be executed in the CVM so schema validation is very important.

Risk: Medium. This requires a previously correct schema definition to be developed. The actual validation can be accomplished with standard API calls.

Constraints:

Usability

- Internal to system.

Reliability

- Mean Time to Failure – 1 failure for every 120 hours of operation is acceptable.

Availability

- System should notify the user of a failure gracefully without aborting the program

Performance

- Really fast. Must take less than .50 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Manasa Bharadwaj

Initiation Date: 10/06/2008

Date Last Modified: 10/06/2008

10.2.1.21. Check Semantic Rules

Use Case ID: T2_CME_SUB_21_CheckSemanticRules

Use Case Level: Functional Sub-use Case

Details

Actors: Modeler

Pre-conditions: Modeler is executing T2_CME_SYSE2E_19_ValidateModel for a model in XCML format. There are a group of previously defined semantic rules saved in the repository.

Description: The CME will check that the current model complies with the defined semantic rules (for an example of the possible set of rules to be checked see "Appendix E – Example of Semantic Rules").

Trigger: The Modeler has started the T2_CME_SYSE2E_19_ValidateModel use case. This use case calls T2_CME_SUB_21_CheckSemanticRules to check that the model complies with the previously defined semantic rules (the calling use case passes the XCML model as a data structure of in-memory CML objects as parameter).

The system responds by …

1. CME reads the saved semantic rules into a custom data structure of rules where each item contains the definition of the rule and a reference to the piece of code that needs to be executed to check if the model complies with the rule (see "Appendix E – Example of Semantic Rules").
2. For each of the rules, CME executes the relevant rule code (the CML data structure is passed as parameter) and accumulates the results in a list.
3. CME returns the results of the validation done in step 2. (Either VALID or INVALID with the list of errors and warnings).

Post-conditions: the calling use case now has the results of the semantic validation.

Alternative courses of action:

None.

Extensions:

None.

Exceptions:

- The schema is not found in the repository; in this case, CME will issue an appropriate error message and raise an exception back to the calling use case.

Related use cases:

- Called by T2_CME_SYSE2E_19_ValidateModel
- Preceded by T2_CME_SUB_20_CheckModelSchema

Decision support

Frequency: Moderate - Performed for every time the Modeler wants to validate an XCML model (10 times per day).

Criticality: High. The end product of CME must be a valid XCML model to be executed in the CVM so semantic validation is very important.

Risk: High. This requires a previously correct set of semantic rules to be developed. It also needs a custom data structure and algorithm to execute the set of rules and return the results to the calling code.

Constraints:

Usability

- Internal to system.

Reliability

- Mean Time to Failure – 1 failure for every 60 hours of operation is acceptable.

Availability

- System should notify the user of a failure gracefully without aborting the program

Performance

- Really fast. Must take less than .50 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.
- The set of rules is read at runtime so that new rules can be added to the repository and the rest of the code does not have to change.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 10/06/2008

Date Last Modified: 10/06/2008

10.2.1.22. Print Model

Use Case ID: T2_CME_SYSE2E_22_PrintModel

Use Case Level: System-Level End-to-End

Details

Actors: Modeler

Pre-conditions: Modeler has the Communication Modeling Environment (CME) open and has loaded the Model he wants to print into the canvas.

Description: The Modeler will print the open model.

Trigger: The Modeler clicks the "Print" button in the tool bar of the CME.

The system responds by …

1. CME presents the Standard Windows Print Dialog.
2. The Modeler accepts the defaults and clicks the "Print" button.
3. The system will then send the printing job to default printer.
4. The default printer finishes printing the model.

Post-conditions: a copy of the model has been printed in the default printer and the model is still open in the editor and ready to be edited.

Alternative courses of action:

At step 2 the user can select any printing preferences from the open dialog box (i.e. number of copies, pages, printer selected, etc.).

Before step 3 the Modeler has the option to cancel the print request by pressing "Cancel" button.

Extensions:

None.

Exceptions:

- The system could not find any printers connected to the workstation.
- The printer could have a paper jam.
- The printer could run out of paper or ink.

Related use cases:

- T2_CME_SYSE2E_05_OpenExistingGCMLModel

Decision support

Frequency: Low. Only when ME user wants to print a hard copy of the model (graphical) about 5 times per day.

Criticality: Low. Printing is not necessary to realize the communications model.

Risk: Low. Straightforward implementation using standard API calls.

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1 failure for every 40 hours of operation is acceptable.

Availability

- Only when model is loaded

Performance

- The model shall be sent to the printer within 5 seconds. Further performance limited by the printer capacity and load (queue of jobs being printed).

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Jorge Jauregui

Initiation Date: 09/18/2008

Date Last Modified: 09/24/2008

10.2.1.23. Edit Model Metadata

Use Case ID: T2_CME_SYSE2E_23_EditModelMetadata

Use Case Level: System-Level End-to-End

Details

Actors: Modeler

Pre-conditions: Modeler has the Communication Modeling Environment (CME) open.

Description: The Modeler will edit the additional metadata that describes an existing model.

Trigger: The Modeler clicks the "Models Administration" menu item in the CME.

The system responds by …

1. CME presents the Models Administration Window.
2. The Modeler selects the desired model from the Models list box.
3. The CME selects the entry in the window.
4. Modeler edits the name, description, and available formats into the appropriate text boxes.
5. The Modeler clicks "Change" to accept the changes made to this entry.
6. The Modeler clicks "Save" button.
7. The CME system saves the modified entry into the repository data structure and saves a copy of the model into the internal repository folder.

Post-conditions: the modified model entry is saved in the repository data structure.

Alternative courses of action:

At steps 2 to 5 the Modeler can hit "Cancel" button to cancel the edition of the entry.

Extensions:

None.

Exceptions:

- The Modeler can enter a name that already exists. The CME will warn the modeler that this action will overwrite the exiting model.

Related use cases:

- T2_CME_SYSE2E_11_AddModelToRepository

Decision support

Frequency: Moderate - Performed for every time the Modeler wants to edit a model entry in the repository (10 times per day).

Criticality: Medium – although requiring more work, the Modeler can delete the entry and add it again to accomplish the same results.

Risk: Medium

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1 failure for every 24 hours of operation is acceptable.

Performance

- The model metadata entry shall be saved within 2 seconds.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 09/18/2008

Date Last Modified: 10/04/2008

10.2.1.24. Login

Use Case ID: T2_CME_SYSE2E_24_Login

Use Case Level: System-Level End-to-End

Details

Actors: User

Pre-conditions: The user has the CME installed and has access to the Repository. He is currently not using the CME.

Description: The user wishes to access the system.

Trigger: The user starts the application by click on the application icon.

The system responds by …

1. CME displays the login dialog prompting the user to enter his credentials.

2. The user types his credentials and clicks OK.
3. The system retrieves the typed information and compares it with the user's credentials stored in the repository.
4. The CME displays the user's main screen.

Post-conditions: After this use case is complete the CME is open and the user is ready to interact with the system.

Alternative courses of action:

At step 2 the user has the option to cancel the login request by pressing "Cancel" button. Then the application is closed.

Extensions:

None.

Exceptions:

- The user can provide erroneous credentials.
- A connection to the repository could not be established.

Related use cases:

- Protects against T2_CME_MISUSE_01_UnauthorizedAccess
- Uses T2_CME_SUB_31_SuspendUserAccountAfterNAttempts
- T2_CME_SYSE2E_25_Logout
- T2_CME_SYSE2E_33_UnlockApplication

Decision support

Frequency: Moderate. This needs to be done every time the system is started (about 8 times per day).

Criticality: High. The CME must be secured so that unauthorized use is prevented.

Risk: Medium. Implementation is easy but must account for misuse so that credentials information must be secured.

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1 failure for every 80 hours of operation is acceptable.

Performance

- User must be able to access the system within 3 seconds of pressing the OK button.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 09/18/2008

Date Last Modified: 09/27/2008

10.2.1.25. Logout

Use Case ID: T2_CME_SYSE2E_25_Logout

Use Case Level: System-Level End-to-End

Details

Actors: User

Pre-conditions: The user has the CME open and running.

Description: The user wishes to exit the system.

Trigger: The user clicks on the "Exit" button.

The system responds by …

1. CME displays the logout dialog prompting the user to confirm the operation or cancel.
2. The user clicks on the OK button confirming the action.
3. The system terminates all processes and closes.

Post-conditions: After this use case is complete the user has left the system.

Alternative courses of action:

At step 1, if the work in progress is not saved, the system displays a dialog box asking if he wants to save with a “Save”, “Don't Save” and “Cancel” buttons.

a- If the user presses “Save”, the work in progress is saved and use case proceeds.

b- If the user presses “Don't save”, the dialog box is closed and use case proceeds.

c- If the user presses “Cancel” the use case ends.

At step 2, the user can decide to cancel the action; this cancels the execution of the use case, in which case he would continue to use the CME.

Extensions:

- None.

Exceptions:

- None.

Related use cases:

- T2_CME_SYSE2E_24_Login
- T2_CME_SYSE2E_32_LockApplication
- Protects against T2_CME_MISUSE_02_UnauthorizedUseOfRunningApplication

Decision support

Frequency: Moderate. This needs to be done every time the system is closed (about 8 times per day).

Criticality: Moderate. The CME must be secured so that unauthorized use is prevented.

Risk: Low. Straightforward implementation.

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1 failure for every 160 hours of operation is acceptable.

Performance

- User must be able to exit the system within 2 seconds of pressing the OK button.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 09/18/2008

Date Last Modified: 09/27/2008

10.2.1.26. Create Mirror Backup of Repository

Use Case ID: T2_CME_SYSE2E_26_CreateMirrorBackupOfRepository

Use Case Level: System-Level End-to-End

Details

Actors: Administrator

Pre-conditions: The administrator has logged into the system and is on the system configuration screen (CME Administrative tool).

Description: The user wishes to set the automatic back up feature.

Trigger: The administrator clicks administrator clicks on the "Repository Settings" option.

The system responds by …

1. CME displays the Repository Settings Window.
2. Under the "Mirror Options", the user clicks on the button "Mirror now".
3. The system copies the contents of the repository to another hidden location (configurable from this same screen).

Post-conditions: After this use case is complete the default repository and the mirror repository contain the exact same information and files.

Alternative courses of action:

At step 2, the administrator has the option to set a recurring time in which the repository will be mirrored (for example, daily).

Extensions:

None.

Exceptions:

- A connection to the repository could not be established.
- The file system throws an error while copying files (i.e. out of space)

Related use cases:

- Protects against T2_CME_MISUSE_01_UnauthorizedAccess

Decision support

Frequency: Low. The system can be configured so that the mirroring can occur once a day at a specific time. The administrator can mirror the repository at any time also. The repository will be mirrored on average once a day.

Criticality: High. The repository data is very valuable and must be protected against malicious or accidental deletion.

Risk: Low. Uses standard system calls to copy the files.

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1 failure for every 160 hours of operation is acceptable.

Availability

- Should always be available to the administrator. Down time of the CME fewer than 30 minutes every 80 hours of operation.

Performance

- User must be able to schedule backup within 2 seconds of clicking the "Mirror Now" option. Time to complete the mirror backup Depends on the computer the system is located in and the time it takes to copy files from one location to another.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Jorge Jauregui

Initiation Date: 09/18/2008

Date Last Modified: 09/30/2008

10.2.1.27. Encrypt Sensitive Data

Use Case ID: T2_CME_SYSE2E_27_EncryptSensitveData

Use Case Level: System-Level End-to-End

Details

Actors: User

Pre-conditions: The administrator has logged into the system and is on the system configuration screen (CME Administrative tool).

Description: The user wishes to set the automatic back up feature.

Trigger: The administrator or a modeler is editing authorized user record in the repository such as creating user or changing the password. After all changes, the user clicks "Save".

The system responds by …

1. CME selects the group of attributes marked as sensitive (for example, user name and password).
2. CME encrypts the selected attributes and returns the list with the corresponding encrypted equivalents.
3. CME saves the records in the repository with the sensitive attribute values replaced by the encrypted equivalents.
4. Now any user or misuser reading the repository files will not be able to decode the sensitive information since it is encrypted.

Post-conditions: Sensitive attribute values are encrypted in the repository file (one-way encryption).

Alternative courses of action:

None.

Extensions:

None.

Exceptions:

- A connection to the repository could not be established.
- The file system throws an error while copying files (i.e. out of space)

Related use cases:

- Protects against T2_CME_MISUSE_04_ReadSensitiveDataInRepository
- Authorize Data Reads through the CME system will decrypt the sensitive data.

Decision support

Frequency: Moderate to low. About 20 times per week

Criticality: High. The repository data is very valuable and must be protected against malicious users.

Risk: Low. Uses standard encryption algorithms

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1 failure for every 160 hours of operation is acceptable.

Availability

- Should always be available to the administrator. Down time of the CME fewer than 30 minutes every 80 hours of operation.

Performance

- CME will take less than .005 seconds to encrypt sensitive data.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Jorge Jauregui

Initiation Date: 09/18/2008

Date Last Modified: 10/08/2008

10.2.1.28. Create Authorized User Account

Use Case ID: T2_CME_SYSE2E_28_CreateAuthorizedUserAccount

Use Case Level: System-Level End-to-End

Details

Actors: Administrator

Pre-conditions: A new user has to be added to the system. The administrator is logged to the system.

Description: The user wishes to access the system.

Trigger: The administrator clicks on the "Add New User" in the CME administrative tool.

The system responds by …

1. CME displays the Add New User dialog box.
2. The administrator enters the user ID, full name, and default password, he also sets the role as Modeler.
3. Administrator clicks Add User.
4. The system saves the new user.
5. The CME displays the Administrative tool main screen.

Post-conditions: After this use case is complete the new user has been saved into the CME repository.

Alternative courses of action:

At step 3 and before, the administrator has the option to cancel the request by pressing "Cancel" button. Then the use case is aborted and CME displays the previous screen.

Extensions:

None.

Exceptions:

- A connection to the repository could not be established.

Related use cases:

- Protects against T2_CME_MISUSE_01_UnauthorizedAccess
- T2_CME_SYSE2E_24_Login
- T2_CME_SYSE2E_30_DeleteUserAccount

Decision support

Frequency: Low. This needs to be done every time a new user is added. About 10 times per month

Criticality: High. The CME must be secured so that unauthorized use is prevented.

Risk: Medium. Implementation is easy but must account for misuse so that credentials information must be secured.

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1 failure for every 160 hours of operation is acceptable.

Performance

- User must be able to access the system within 3 seconds of pressing the Add User button.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Sandeep Varry

Initiation Date: 09/18/2008

Date Last Modified: 09/27/2008

10.2.1.29. Change User Account Password

Use Case ID: T2_CME_SYSE2E_29_ChangeUserAccountPassword

Use Case Level: System-Level End-to-End

Details

Actors: User

Pre-conditions: A new user has been added to the system.

Description: The user wishes to access the system.

Trigger: The user logs into the system for the first time as specified in Login use case.

The system responds by …

1. CME displays a "First Time login change password" dialog box.
2. The user enters the "old password", "new password" and "confirm new password" values in the appropriate text boxes.
3. User clicks Change Password button.
4. The system saves the new password.
5. The CME displays the main screen.

Post-conditions: After this use case is complete the new password has been saved to the repository.

Alternative courses of action:

The trigger can also be that the user clicks the Change password button in the CME main window.

The CME system can also be configured by an administrator so that it prompts to change the password after certain amount of time (for example, every 30 days).

At step 3, the CME can issue an error back to the user and prompts for the password again if new password and confirm new password are different or if old password is equal to new password.

Extensions:

None.

Exceptions:

- A connection to the repository could not be established.

Related use cases:

- Protects against T2_CME_MISUSE_05_AccessSystemWithStolenCredentials
- T2_CME_SYSE2E_24_Login

Decision support

Frequency: Moderate. This needs to be done every time a new user is added or when the user requests it. It will be executed about 30 times per month (once per user). The administrator can set a time period before the password has to be changed.

Criticality: High. The CME must be secured so that unauthorized use is prevented.

Risk: Medium. Implementation is easy but must account for misuse so that credentials information must be secured.

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1 failure for every 160 hours of operation is acceptable.

Performance

- User must be able to access the system within 3 seconds of pressing the Change Password button.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Manasa Bharadwaj

Initiation Date: 09/18/2008

Date Last Modified: 10/08/2008

10.2.1.30. Delete User Account

Use Case ID: T2_CME_SYSE2E_30_DeleteUserAccount

Use Case Level: System-Level End-to-End

Details

Actors: Administrator

Pre-conditions: A user has to be deleted from the system. The administrator is logged to the system.

Description: The administrator wishes to revoke access the system to a user.

Trigger: The administrator selects the user he wants to delete in the CME administrative tool.

The system responds by …

1. CME selects the user and shows the user details.
2. Administrator clicks on the "Delete This User".
3. CME deletes the account; the system no longer recognizes the user account as valid; the user will not be allowed to log back into the system; the user account will be permanently deleted.
4. The CME displays the Administrative tool main screen.

Post-conditions: After this use case is complete the user account has been permanently deleted from the CME repository.

Alternative courses of action:

At step 2 and before, the administrator has the option to cancel the request by pressing "Cancel" button. Then the use case is aborted and CME displays the previous screen.

Extensions:

None.

Exceptions:

- A connection to the repository could not be established.

Related use cases:

- Protects against T2_CME_MISUSE_06_AccessSystemWithExpiredCredentials
- T2_CME_SYSE2E_24_Login
- T2_CME_SYSE2E_28_CreateAuthorizedUserAccount

Decision support

Frequency: Low. This needs to be done every time a new user is added. About 10 times per month

Criticality: High. The CME must be secured so that unauthorized use is prevented.

Risk: Medium. Implementation is easy but must account for misuse so that credentials information must be secured.

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1 failure for every 160 hours of operation is acceptable.

Performance

- User must be able to access the system within 3 seconds of pressing the Add User button.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Manasa Bharadwaj

Initiation Date: 09/18/2008

Date Last Modified: 09/27/2008

10.2.1.31. Suspend User Account after n attempts

Use Case ID: T2_CME_SUB_31_SuspendUserAccountAfterNAttempts

Use Case Level: Functional Sub-use Case

Details

Actors: User / Hacker

Pre-conditions: The administrator has set up the MaxLoginAttempts configuration setting (n <= [3..7]).

Description: The user / hacker wishes to access the system. As part of the Login use case the CME keeps a count of how many times somebody has tried to access the system unsuccessfully.

Trigger: After entering credentials the user / hacker clicks OK (see Login use case).

The system responds by …

1. The System does not allow the user / hacker to login as he entered wrong credentials.
2. CME displays error message and informs how many attempts left before suspending account.
3. The user or hacker enters different password and then he clicks OK.
4. Previous steps are repeated n times (n = MaxLoginAttempts).
5. The CME suspends the user account and does not allow any more attempts to login.

Post-conditions: The user or hacker is not able to get access the system. The account has been suspended and only an administrator can restore access to that account.

Alternative courses of action:

At step 2 the hacker enters a different user id, and then the CME starts counting attempts for the new id.

Extensions:

- None.

Exceptions:

- A connection to the repository could not be established.

Related use cases:

- T2_CME_SYSE2E_24_Login
- Protects against T2_CME_MISUSE_07_AccessSystemAfterManyAttempts

Decision support

Frequency: Moderate. This needs to be done every time the system is started (about 8 times per day).

Criticality: High. The CME must be secured so that unauthorized use is prevented.

Risk: Medium. Implementation is easy but must account for misuse so that credentials information must be secured.

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1 failure for every 160 hours of operation is acceptable.

Performance

- User must be able to access the system within 3 seconds of pressing the OK button.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Sandeep Varry

Initiation Date: 09/18/2008

Date Last Modified: 09/28/2008

10.2.1.32. Lock Running Application

Use Case ID: T2_CME_SYSE2E_32_LockRunningApplication

Use Case Level: System-Level End-to-End

Details

Actors: User

Pre-conditions: The user has the CME open and running.

Description: The user wishes to lock the system without closing it.

Trigger: The user clicks on the “Lock” button.

The system responds by …

1. CME displays the Lock dialog prompting the user to confirm the operation or cancel.
2. The user clicks on the OK button confirming the action.
3. The system hides the main window and presents the “Unlock CME” Dialog box instead.

Post-conditions: After this use case is complete the main window is hidden and the only available function is a dialog box to unlock the application.

Alternative courses of action:

At step 1, if the work in progress is not saved, the system displays a dialog box asking if he wants to save with a “Save”, “Don't Save” and “Cancel” buttons.

d- If the user presses “Save”, the work in progress is saved and use case proceeds.

e- If the user presses “Don't save”, the dialog box is closed and use case proceeds.

f- If the user presses “Cancel” the use case ends.

At step 2, the user can decide to cancel the action; this cancels the execution of the use case, in which case he would continue to use the CME.

The CME can be configured by an administrator so that lock is automatic after a certain amount of time of being inactive (for example, after 10 minutes of inactivity).

Extensions:

- Extended by T2_CME_SYSE2E_33_UnlockApplication

Exceptions:

- None.

Related use cases:

- T2_CME_SYSE2E_25_Logout
- Protects against T2_CME_MISUSE_02_UnauthorizedUseOfRunningApplication
- Extended by T2_CME_SYSE2E_33_UnlockApplication

Decision support

Frequency: Moderate. This needs to be done every time the system is locked (about 8 times per day).

Criticality: Moderate. The CME must be secured so that unauthorized use is prevented.

Risk: Low. Straightforward implementation.

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1 failure for every 160 hours of operation is acceptable.

Performance

- User must be able to exit the system within 2 seconds of pressing the OK button.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:

Owner: Lazaro Pi

Initiation Date: 09/18/2008

Date Last Modified: 10/08/2008

10.2.1.33. Unlock Application

Use Case ID: T2_CME_SYSE2E_33_UnlockApplication

Use Case Level: System-Level End-to-End

Details

Actors: User

Pre-conditions: The user previously locked the CME.

Description: The user wishes to unlock the system.

Trigger: The user clicks on the "Unlock" button in the "Unlock CME" dialog box.

The system responds by …

1. Presenting a variation of the login dialog box where the user name is read only and the user must enter the password. The steps to authorize the user back are similar to login use case.
2. The user enters password and clicks OK.
3. The system shows the previously hidden main window so that the user can resume work.

Post-conditions: After this use case is complete the main window is restored an the user is back where he left before locking the application.

Alternative courses of action:

At step 2, the user can enter a wrong password; this behaves exactly as explained in the login use case.

Extensions:

- Extends T2_CME_SYSE2E_32_LockRunningApplication

Exceptions:

- None.

Related use cases:

- T2_CME_SYSE2E_24_Login
- Protects against T2_CME_MISUSE_02_UnauthorizedUseOfRunningApplication
- Extends T2_CME_SYSE2E_32_LockRunningApplication

Decision support

Frequency: Moderate. This needs to be done every time the system is locked (about 8 times per day).

Criticality: Moderate. The CME must be secured so that unauthorized use is prevented.

Risk: Low. Straightforward implementation.

Constraints:

Usability

- No previous training is needed.

Reliability

- Mean Time to Failure – 1 failure for every 160 hours of operation is acceptable.

Performance

- User must be able to exit the system within 2 seconds of pressing the OK button.

Supportability

- The application will rely on the Java platform (version 5 or later) so it can be ported to any environment where Java can be installed.

Implementation

- Client requests the implementation to be done in Java and the Eclipse framework.

Modification History:
Owner: Lazaro Pi
Initiation Date: 09/18/2008
Date Last Modified: 10/08/2008

10.2.2. Misuse Case Descriptions

The misuse cases identify security threats against the system. Use cases to prevent those threats have been presented in the previous section.

The relation between the misuse cases and the security use cases described in previous section is expressed in the "Related use cases" subtitle of the use case description. In the misuse case there is an entry preceded with "Prevented by …"; In the security use case there is an entry preceded by "Protects against …".

The same conventions apply for the creation of the misuse case ID's as for the use cases. The only difference is the addition of the MISUSE tag to identify this as a misuse case.

10.2.2.1. Unauthorized Access

Use Case ID: T2_CME_MISUSE_01_UnauthorizedAccess
Use Case Level: Misuse Case

Details

Actors: Hacker

Pre-conditions: The hacker has gained access to a computer where the CME is installed.

Description: Hacker wants to use the CME application.

Trigger: The Hacker clicks on the application icon in order to run the application.

The system responds by …

1. The System does not perform any authentication.
2. The CME displays the user's main screen.

Post-conditions: After this misuse case is complete the CME is open and the hacker is ready to interact with the system.

Alternative courses of action:

- None.

Extensions:

- None.

Exceptions:

- A connection to the repository could not be established.

Related use cases:

- Prevented by T2_CME_SYSE2E_23_Login
- Prevented by T2_CME_SYSE2E_27_CreateAuthorizedUserAccount

Decision support

Frequency: Moderate. Hackers attempt this misuse around 20 times per week.

Criticality: High. The CME must be secured so that unauthorized use is prevented.

Risk: Medium.

Modification History:

Owner: Lazaro Pi

Initiation Date: 09/18/2008

Date Last Modified: 09/28/2008

10.2.2.2. Unauthorized Use of Running Application

Use Case ID: T2_CME_MISUSE_02_UnauthorizedUseOfRunningApplication

Use Case Level: Misuse Case

Details

Actors: Hacker

Pre-conditions: The hacker has gained access to a computer where the CME is installed. The CME application was left open and running by a user.

Description: Hacker wants to use the CME application.

Trigger: The Hacker clicks on window representing the CME.

The system responds by …

1. The CME displays the user's main screen.

Post-conditions: After this misuse case is complete the CME is open and in focus and the hacker is ready to interact with the system.

Alternative courses of action:

- None.

Extensions:

- None.

Exceptions:

- None.

Related use cases:

- Prevented by T2_CME_SYSE2E_24_Logout
- Prevented by T2_CME_SYSE2E_31_LockRunningApplication

Decision support

Frequency: Moderate. Hackers attempt this misuse around 20 times per week.

Criticality: High. The CME must be secured so that unauthorized use is prevented.

Risk: Medium.

Modification History:

Owner: Lazaro Pi

Initiation Date: 09/18/2008

Date Last Modified: 09/28/2008

10.2.2.3. Delete Data in Repository

Use Case ID: T2_CME_MISUSE_03_DeleteDataInRepository

Use Case Level: Misuse Case

Details

Actors: Hacker

Security Threat: All work done on pre-existing communication models can easily be deleted with no way of recovery.

Pre-conditions: The Hacker has gained access to the repository.

Description: Attempted deletion of all X-CML and G-CML files in the repository.

Trigger: The Hacker highlights (selects) all the files in the repository and drags them into the recycling bin / trash.

The system responds by …

1. All files are sent to the Trash.
2. Hacker empties the recycling bin / trash.
3. The system deletes the files from the Trash folder.

Post-conditions: After this misuse case is complete all repository files are deleted and the repository is empty. The users of the CME system cannot recover the deleted files.

Alternative courses of action:

- None.

Extensions:

- None.

Exceptions:

- None.

Related use cases:

- Prevented by T2_CME_SYSE2E_26_CreateMirrorBackupOfRepository

Decision support

Frequency: Moderate. Hackers attempt this misuse around 10 times per week.

Criticality: High. The CME must protect Repository data.

Risk: Medium.

Modification History:

Owner: Jorge Jauregui

Initiation Date: 09/18/2008

Date Last Modified: 09/30/2008

10.2.2.4. Read Sensitive Data in Repository

Use Case ID: T2_CME_MISUSE_04_ReadSensitiveDataInRepository

Use Case Level: Misuse Case

Details

Actors: Hacker

Security Threat: Misuser has all the access privileges of a valid user and can perform actions on behalf of that user.

Pre-conditions: The Hacker has gained access to the repository.

Description: Hacker obtains sensitive data such as user name and passwords from files in the repository.

Trigger: The Hacker clicks open for all the files in the repository.

The system responds by …

1. Files are open in a text editor.
2. For each of the files, Hacker searches through the file for a username and password.
3. When he finds the appropriate attributes, hacker copies the user name and password.
4. Hacker enters this information on the logon screen of the CME.
5. CME grants access to the Hacker and lets him log in.

Post-conditions: After this misuse case is complete the application positively authenticates the Hacker as a valid user and gives him/her access to the system.

Alternative courses of action:

- None.

Extensions:

- None.

Exceptions:

- None.

Related use cases:

- Prevented by T2_CME_SYSE2E_27_EncryptSensitveData

Decision support

Frequency: Moderate. Hackers attempt this misuse around 10 times per week.

Criticality: High. The CME must protect Repository data.

Risk: Medium.

Modification History:

Owner: Jorge Jauregui

Initiation Date: 09/18/2008

Date Last Modified: 10/08/2008

10.2.2.5. Access System with Stolen Credentials

Use Case ID: T2_CME_MISUSE_05_AccessSystemWithStolenCredentials

Use Case Level: Misuse Case

Details

Actors: Hacker

Pre-conditions: The hacker has obtained valid user credentials and has access to a computer where the CME is installed. The hacker has a username and password which he/she has gained through some unknown means (social engineering, etc.).

Security Threat: Hacker has all the access privileges of a valid user and can perform actions on behalf of that user.

Description: Hacker wants to use the CME application.

Trigger: The Hacker clicks on the application icon in order to run the application.

The system responds by …

1. The System performs authentication as explained in login use case.
2. Hacker is given access since the credentials he supplies are valid.
3. The CME displays the user's main screen.

Post-conditions: The Hackers is allowed to login to some authorized users account. After this misuse case is complete the CME is open and the hacker is ready to interact with the system.

Alternative courses of action:

- None.

Extensions:

- None.

Exceptions:

- A connection to the repository could not be established.

Related use cases:

- Prevented by T2_CME_SYSE2E_29_ChangeUserAccountPassword

Decision support

Frequency: Moderate. Hackers attempt this misuse around 20 times per week.

Criticality: High. The CME must be secured so that unauthorized use is prevented.

Risk: Medium.

Modification History:

Owner: Lazaro Pi, Sandeep Varry, Jorge Jauregui

Initiation Date: 09/18/2008

Date Last Modified: 10/08/2008

10.2.2.6. Access System with Expired Credentials

Use Case ID: T2_CME_MISUSE_06_AccessSystemWithExpiredCredentials

Use Case Level: Misuse Case

Details

Actors: Hacker

Pre-conditions: The hacker has obtained expired user credentials and has access to a computer where the CME is installed. For example, the hacker can be a fired employee that has not been deleted from the list of authorized users.

Security Threat: Hacker has all the access privileges of a user.

Description: Hacker wants to use the CME application.

Trigger: The Hacker clicks on the application icon in order to run the application.

The system responds by …

1. The System performs authentication as explained in login use case.
2. Hacker is given access since the credentials he supplies are valid.
3. The CME displays the user's main screen.

Post-conditions: After this misuse case is complete the CME is open and the hacker is ready to interact with the system.

Alternative courses of action:

- None.

Extensions:

- None.

Exceptions:

- A connection to the repository could not be established.

Related use cases:

- Prevented by T2_CME_SYSE2E_30_DeleteUserAccount

Decision support

Frequency: Moderate. Hackers attempt this misuse around 20 times per week.

Criticality: High. The CME must be secured so that unauthorized use is prevented.

Risk: Medium.

Modification History:

Owner: Lazaro Pi, Sandeep Varry, Jorge Jauregui

Initiation Date: 09/18/2008

Date Last Modified: 10/08/2008

10.2.2.7. Access System after Many Login Attempts

Use Case ID: T2_CME_MISUSE_07_AccessSystemAfterManyAttempts

Use Case Level: Misuse Case

Details

Actors: Hacker

Pre-conditions: The hacker has gained access to a computer where the CME is installed.

Description: Hacker tries repeatedly to login to one of the authorized users account.

Trigger: The Hacker enters user id and password and tries to login to one of the registered users account (see Login use case).

The system responds by …

1. The System does not allow the hacker to login as he entered wrong credentials.
2. CME displays error message.
3. The hacker enters different id and/or password and then he clicks OK.
4. Previous steps are repeated until the hacker enters correct id and password and gets access to the system.

Post-conditions: The hacker successfully logs into some authorized users account.

Alternative courses of action:

The hacker can use a malicious application to generate different ID's and passwords to automate the attack on the system.

Extensions:

- None.

Exceptions:

- A connection to the repository could not be established.

Related use cases:

- T2_CME_SYSE2E_23_Login
- Prevented by T2_CME_SUB_30_SuspendUserAccountAfterNAttempts

Decision support

Frequency: Low. Hackers attempt this misuse around 10 times per week.

Criticality: High. The CME must be secured so that unauthorized use is prevented.

Risk: Medium. Implementation is easy but must account for misuse so that credentials information must be secured.

Modification History:

Owner: Sandeep Varry

Initiation Date: 09/18/2008

Date Last Modified: 09/28/2008

10.3. Appendix C – User Interface Design

Login popup window

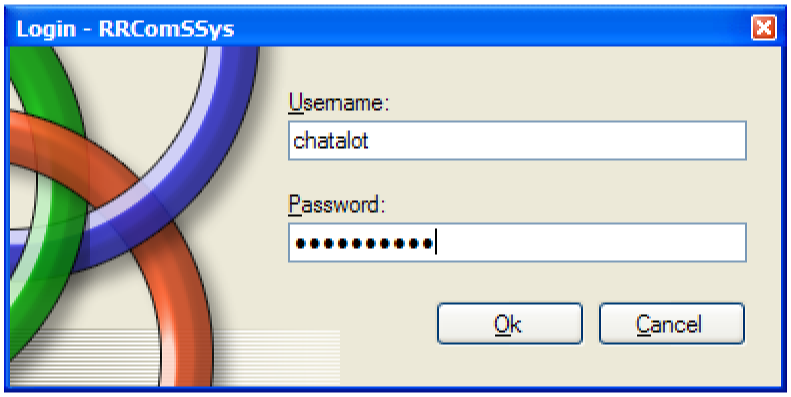
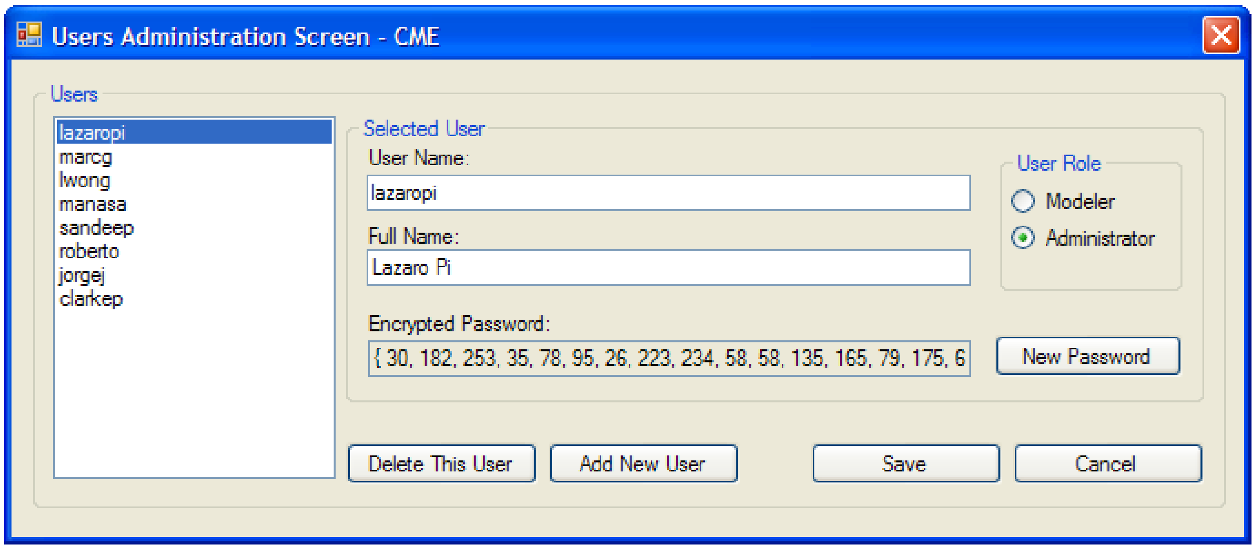


Figure 30 UI, Login

Administrative Window for editing Users


Users Administration Screen - CME

Users

- lazaropi
- marcg
- lwong
- manasa
- sandeep
- roberto
- jorgej
- clarkep

Selected User

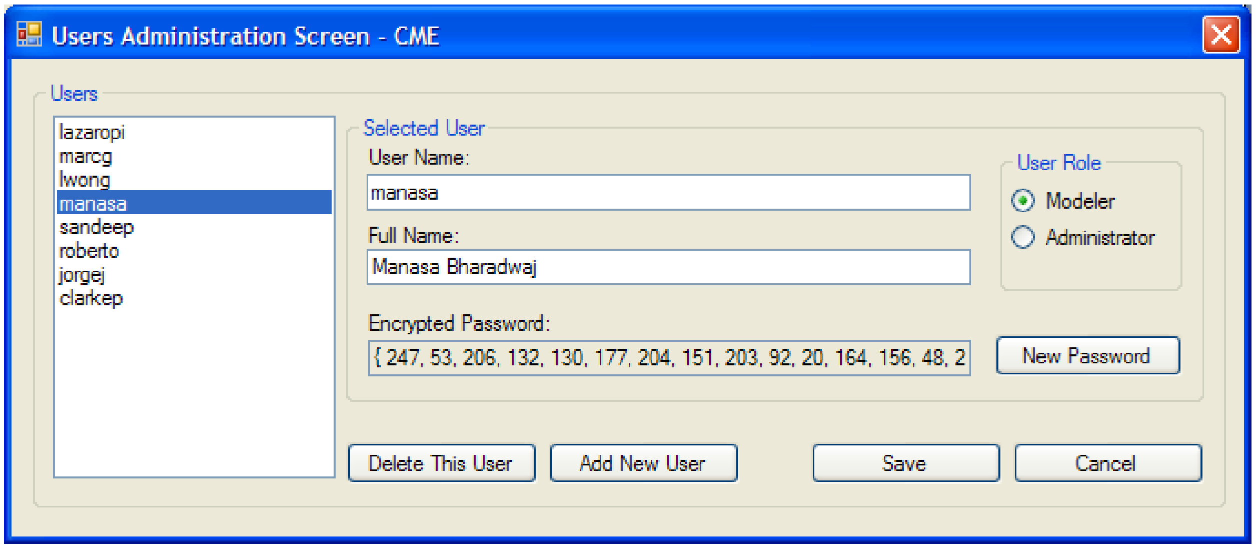
User Name: lazaropi

Full Name: Lazaro Pi

Encrypted Password: { 30, 182, 253, 35, 78, 95, 26, 223, 234, 58, 58, 135, 165, 79, 175, 6

User Role: Modeler [✓ Administrator]

Figure 31 UI, Administrative Window for editing users


Users Administration Screen - CME

Users

- lazaropi
- marcg
- lwong
- manasa
- sandeep
- roberto
- jorgej
- clarkep

Selected User

User Name: manasa

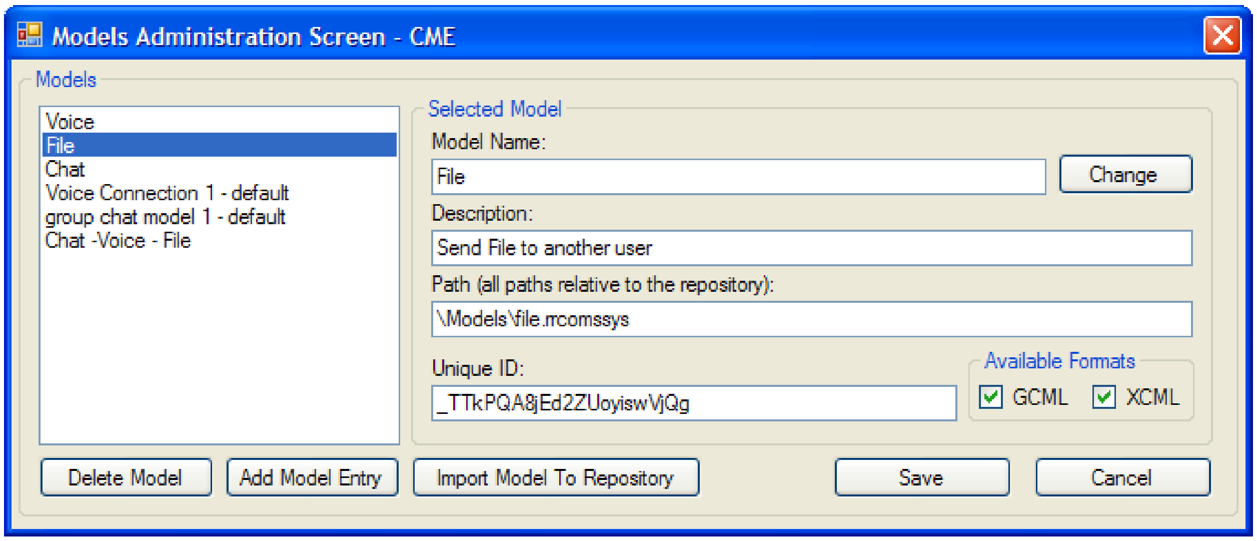
Full Name: Manasa Bharadwaj

Encrypted Password: { 247, 53, 206, 132, 130, 177, 204, 151, 203, 92, 20, 164, 156, 48, 2

User Role: [✓ Modeler] Administrator

Figure 32 UI, editing users

Administrative Window for editing Models Metadata



Models Administration Screen - CME

Models

- Voice
- File
- Chat
- Voice Connection 1 - default
- group chat model 1 - default
- Chat -Voice - File

Selected Model

Model Name: File

Description: Send File to another user

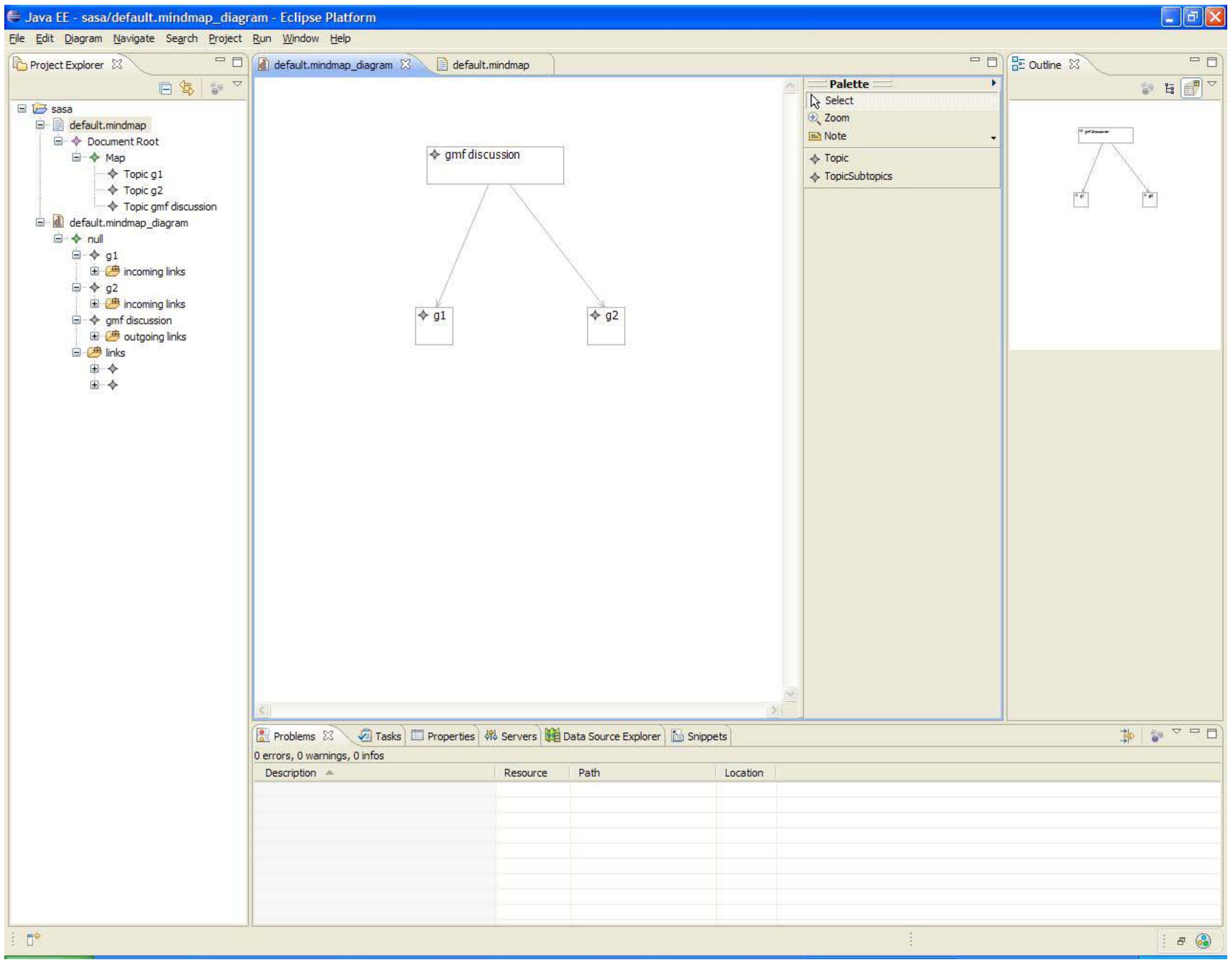
Path (all paths relative to the repository): \Models\file.rcomssys

Unique ID: _TTkPQA8jEd2ZUoyiswVjQg

Available Formats: [✓ GCML] [✓ XCML]

Figure 33 UI, editing metadata

Model for with simple lines and shapes



0 errors, 0 warnings, 0 infos

Figure 34 Model with simple lines and shapes

Model for CML Model for a 2-way Voice connection

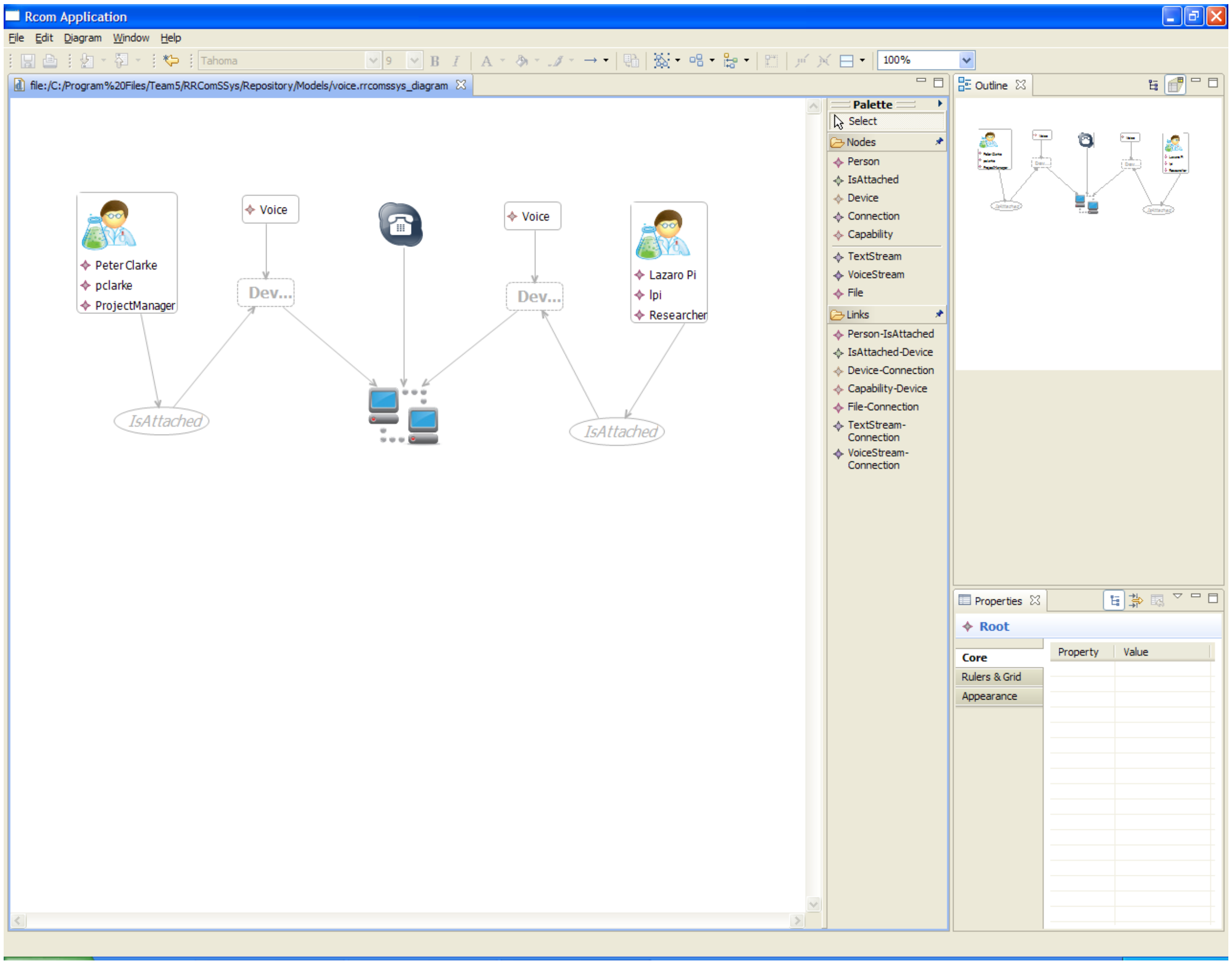


Figure 35 UI, Model for CML model for a 2-way voice connection

Create new model

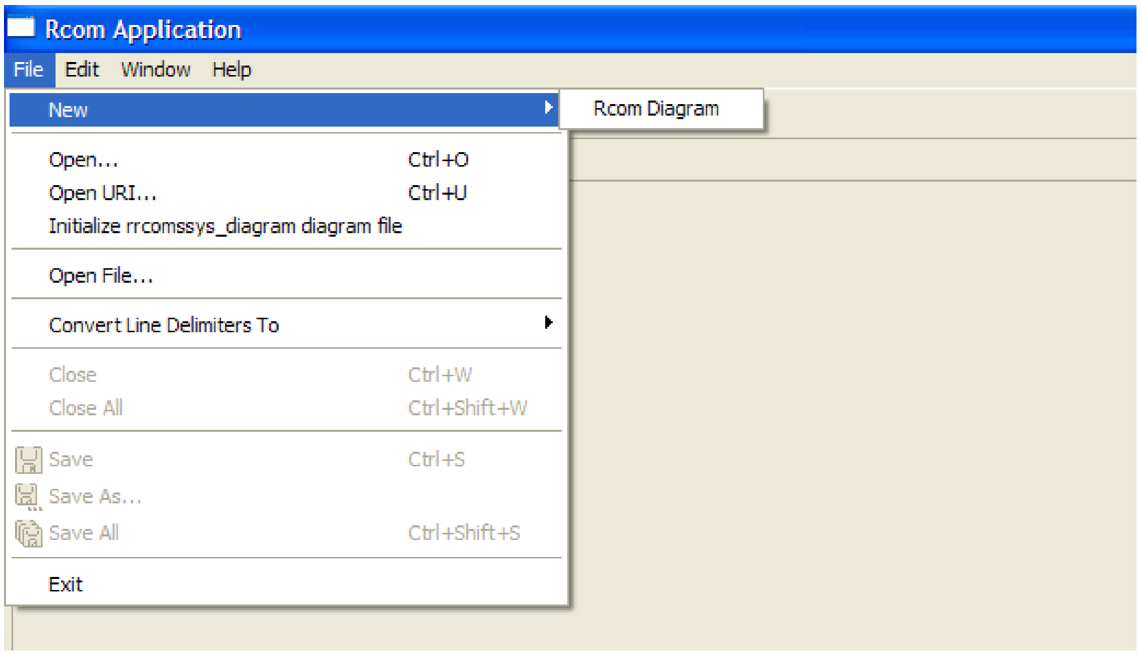
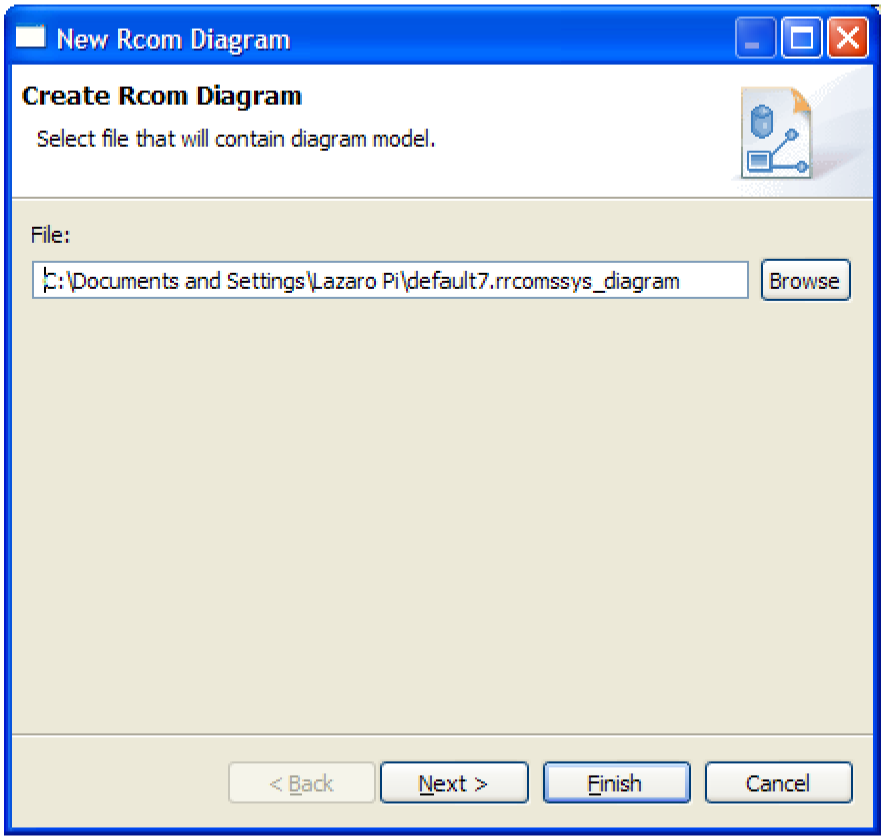


Figure 36 UI, create new model

Enter name for new model



New Rcom Diagram

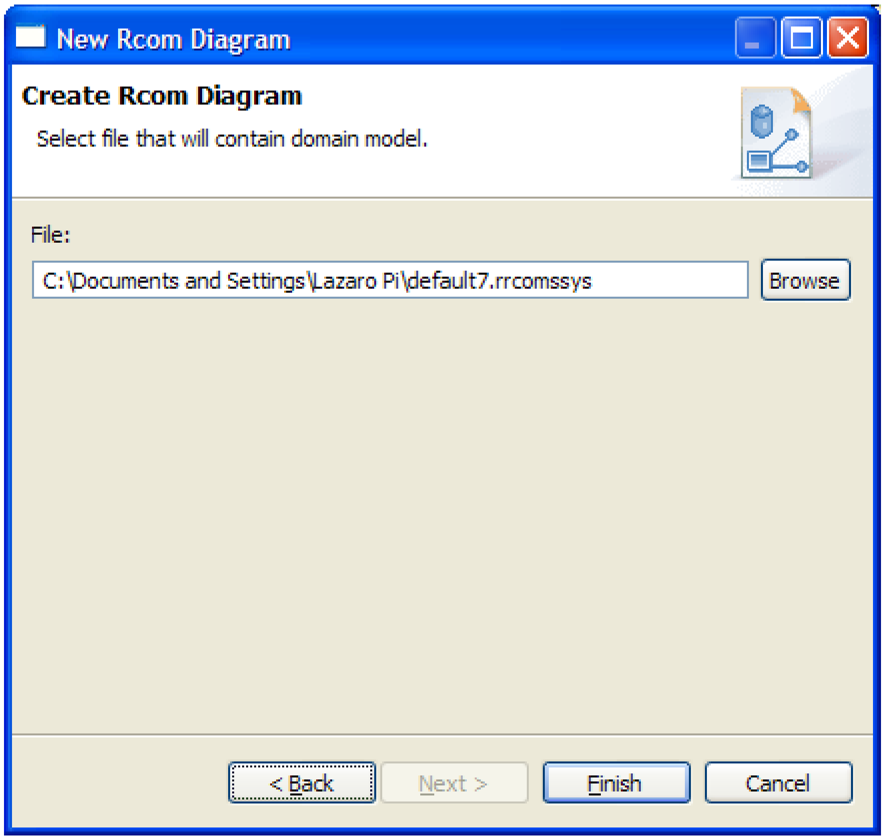
Create Rcom Diagram

Select file that will contain diagram model.

File: C:\Documents and Settings\Lazaro Pi\default7.rrcomssys_diagram

Figure 37 UI, enter name for new model

Enter name for new model 2



Create Rcom Diagram

Select file that will contain domain model.

File:

C:\Documents and Settings\Lazaro Pi\default7.rrcomssys

Figure 38 UI, enter name for new model 2

New model created

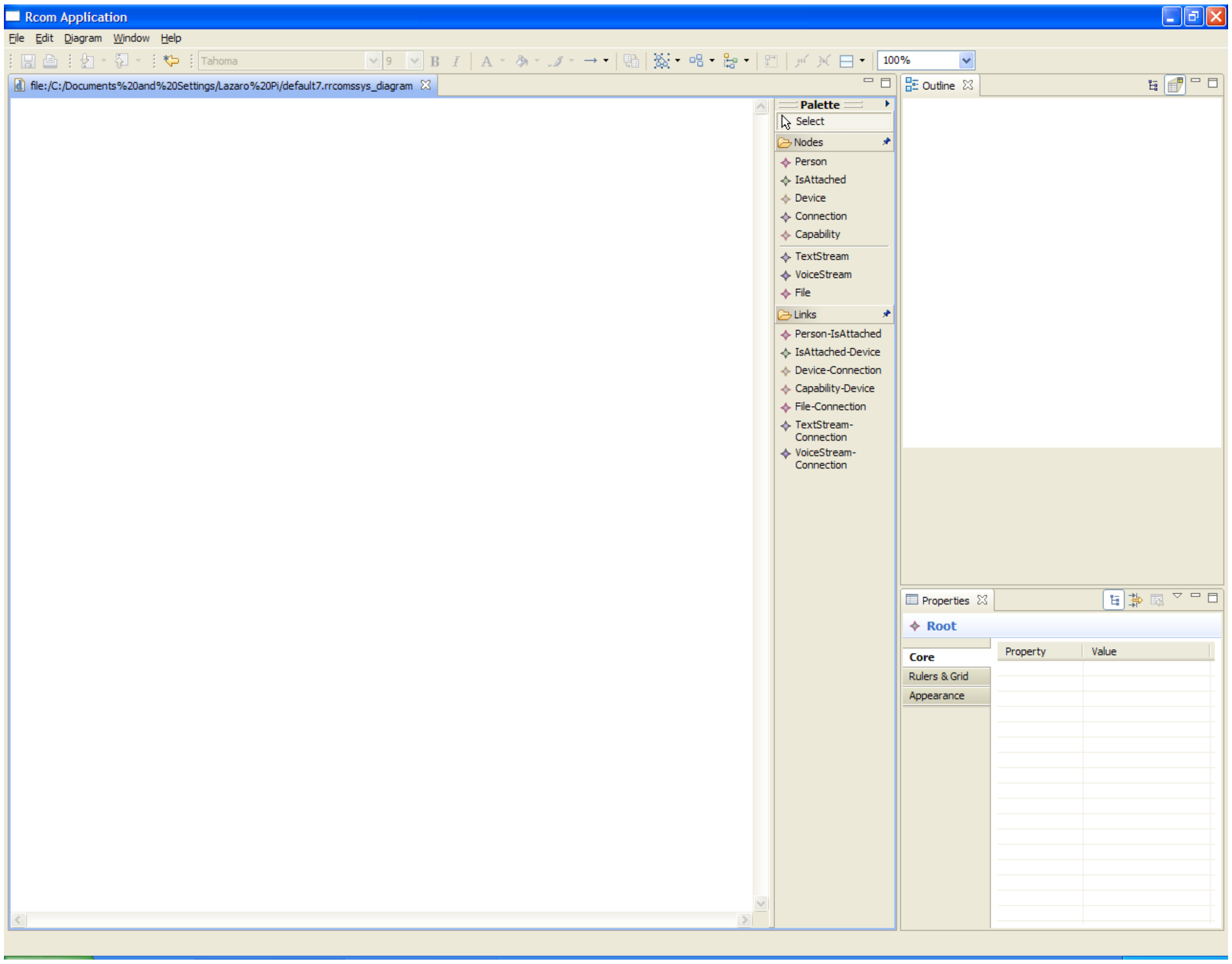
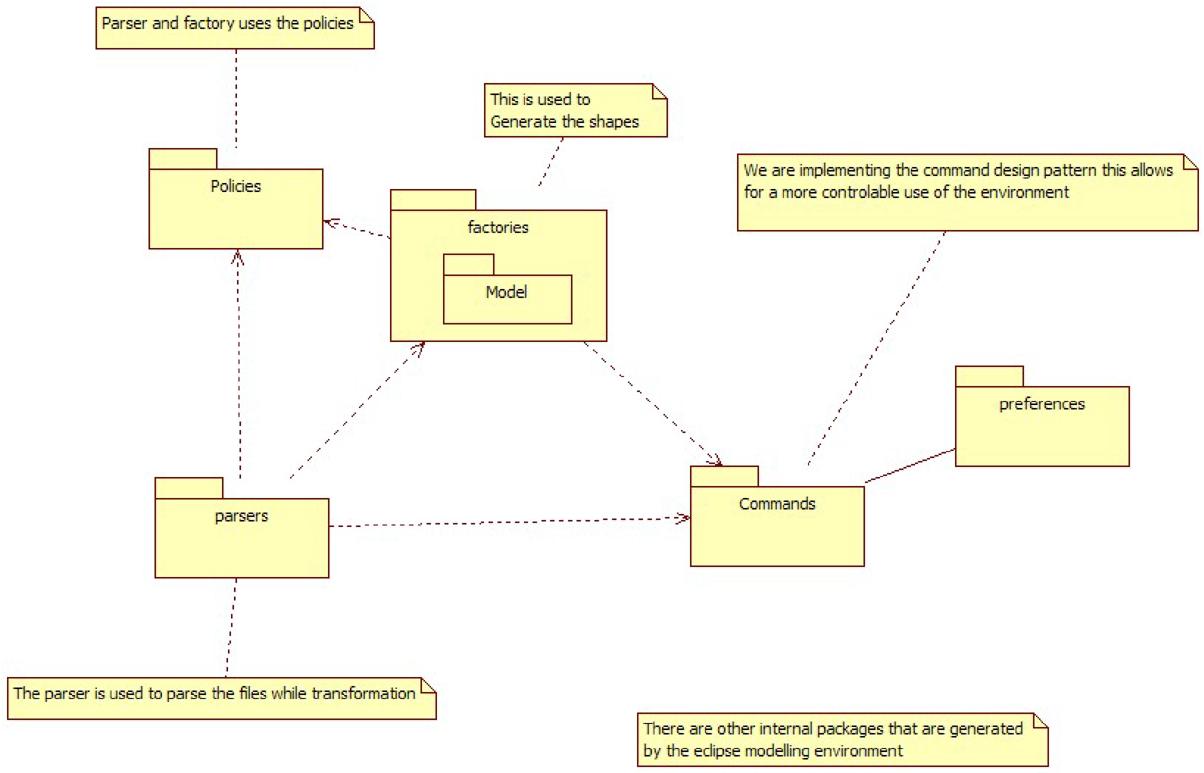


Figure 39 UI, new model created

10.4. Appendix D – Detailed Class Diagrams

CmeController



Parser and factory uses the policies
This is used to Generate the shapes
We are implementing the command design pattern this allows for a more controlable use of the environment
Policies
factories
Model
preferences
parsers
Commands
The parser is used to parse the files while transformation
There are other internal packages that are generated by the eclipse modelling environment

Figure 40 Detailed Class Diagram, CmeController

Gcml

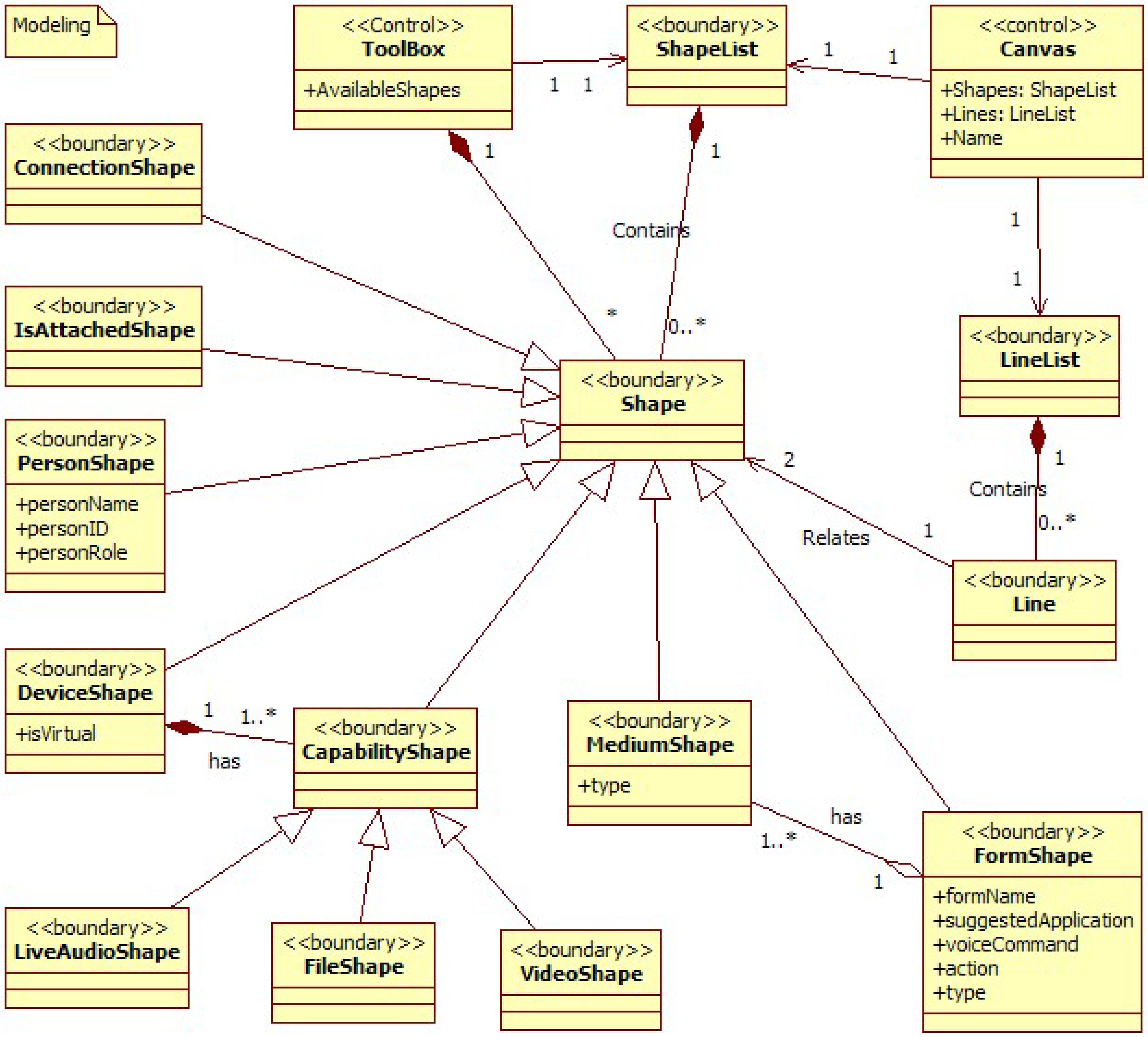


Figure 41 Detailed class diagram, GCML

UCI

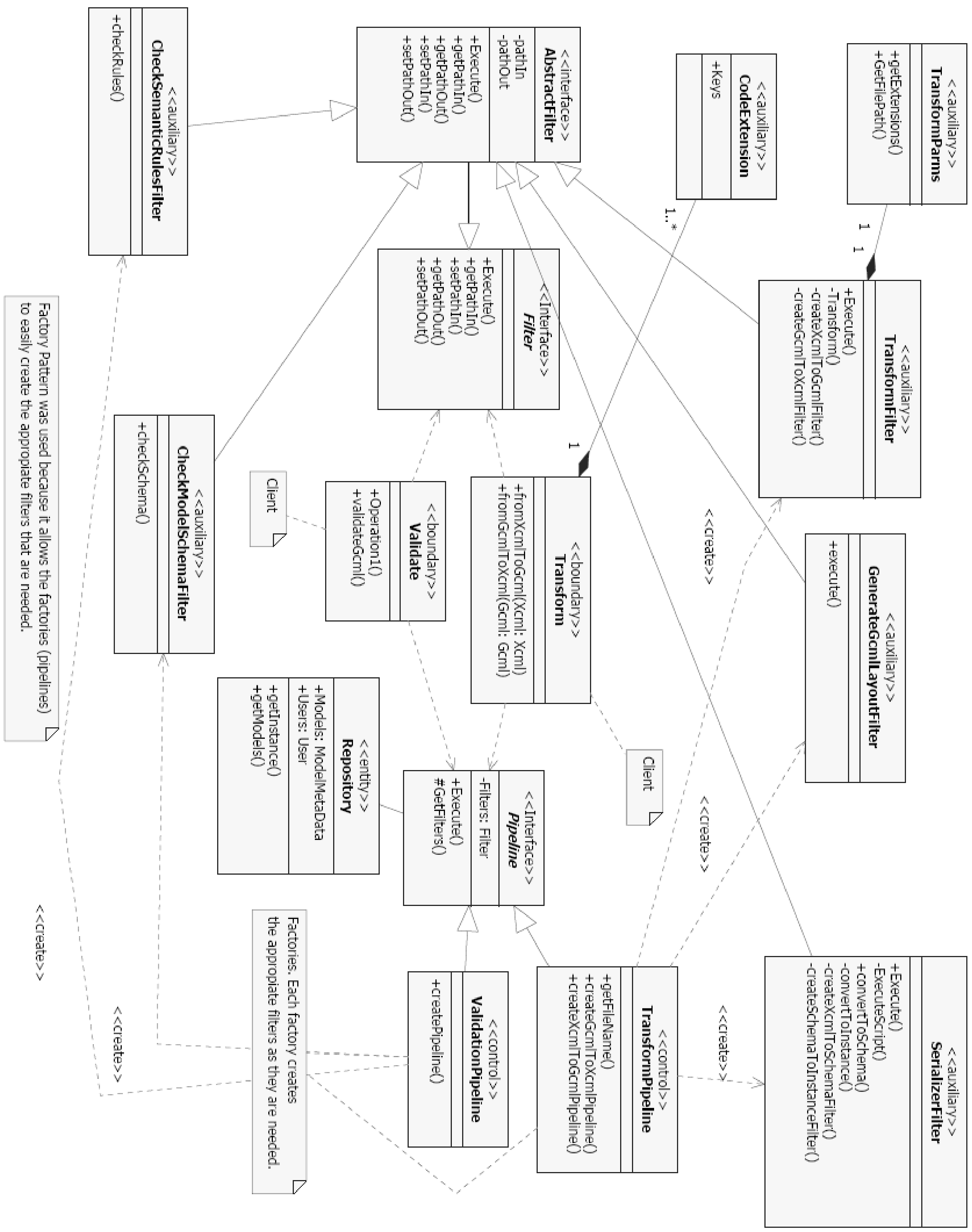


Figure 42 Detailed Class Diagram, UCI

Repository

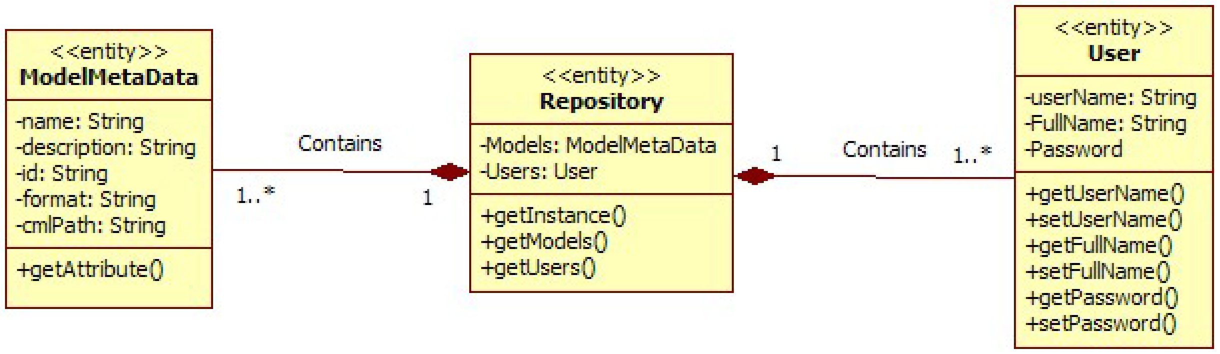


Figure 43 Detailed Class Diagram, Repository

Xcml

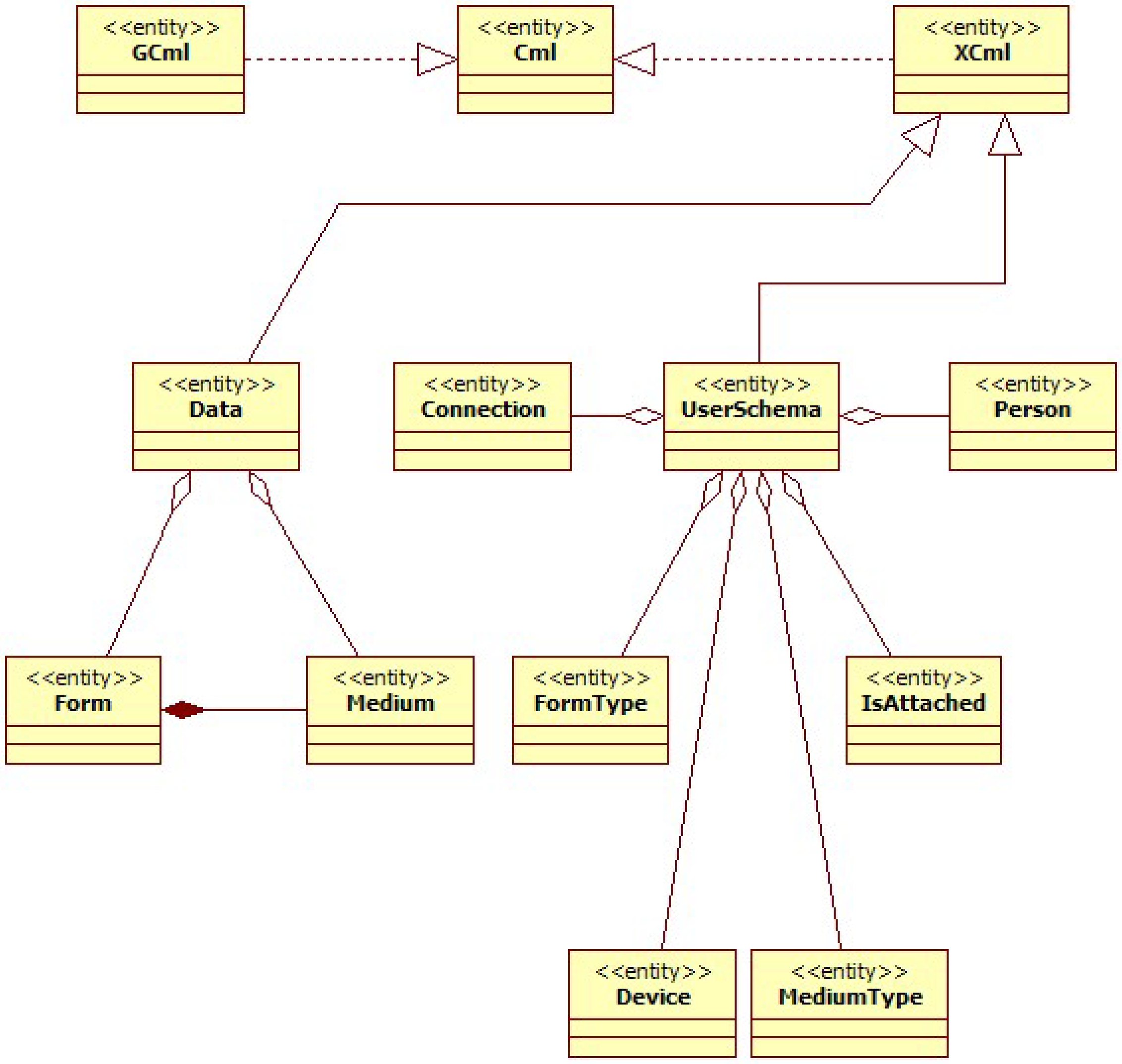


Figure 44 Detailed Class Diagram, XCML

10.5. Appendix E – Class Interfaces

Modeling Environment (gcml.diagram and gcml.edit)

```
package gcml.diagram.edit.commands;

/**
 * @generated
 */
public class ConnectionCreateCommand extends CreateElementCommand {

}

package gcml.diagram.edit.helpers;

/**
 * @generated
 */
public class ConnectionEditHelper extends GcmlBaseEditHelper {
}

package gcml.diagram.edit.parts;

/**
 * @generated
 */
public class ConnectionEditPart extends ShapeNodeEditPart {
}

package gcml.diagram.edit.policies;

/**
 * @generated
 */
public class ConnectionItemSemanticEditPolicy extends
                        GcmlBaseItemSemanticEditPolicy {
}

package gcml.diagram.navigator;

/**
 * @generated
 */
public abstract class GcmlAbstractNavigatorItem extends PlatformObject {
}

package gcml.diagram.parsers;

/**
 * @generated
 */
public abstract class AbstractParser implements IParser {
}

package gcml.diagram.part;

/**
 * @generated
 */
public class DeleteElementAction extends AbstractDeleteFromAction {
}
```

...Many more packages and classes that are part of gcml.diagram and gceml.edit...

Gcml Object Model

```
package gcml;

import java.util.Arrays;
import java.util.Collections;
import java.util.List;

import org.eclipse.emf.common.util.Enumerator;

/**
 * <!-- begin-user-doc -->
 * A representation of the literals of the enumeration '<em><b>Action</b></em>',
 * and utility methods for working with them.
 * <!-- end-user-doc -->
 * @see gcml.GcmlPackage#getAction()
 * @model
 * @generated
 */
public enum Action implements Enumerator {
	/**
	 * The '<em><b>Send</b></em>' literal object.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @see #SEND_VALUE
	 * @generated
	 * @ordered
	 */
	SEND(0, "send", "send"),

	/**
	 * The '<em><b>Do Not Send</b></em>' literal object.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @see #DO_NOT_SEND_VALUE
	 * @generated
	 * @ordered
	 */
	DO_NOT_SEND(1, "doNotSend", "doNotSend"),

	/**
	 * The '<em><b>Start</b></em>' literal object.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @see #START_VALUE
	 * @generated
	 * @ordered
	 */
	START(2, "start", "start");

	/**
	 * The '<em><b>Send</b></em>' literal value.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of '<em><b>Send</b></em>' literal object isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @see #SEND
 * @model name="send"
 * @generated
 * @ordered
 */
public static final int SEND_VALUE = 0;

/**
 * The '<em><b>Do Not Send</b></em>' literal value.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of '<em><b>Do Not Send</b></em>' literal object isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @see #DO_NOT_SEND
 * @model name="doNotSend"
 * @generated
 * @ordered
 */
public static final int DO_NOT_SEND_VALUE = 1;

/**
 * The '<em><b>Start</b></em>' literal value.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of '<em><b>Start</b></em>' literal object isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @see #START
 * @model name="start"
 * @generated
 * @ordered
 */
public static final int START_VALUE = 2;

/**
 * An array of all the '<em><b>Action</b></em>' enumerators.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
private static final Action[] VALUES_ARRAY =
		new Action[] {
			SEND,
			DO_NOT_SEND,
			START,
		};

/**
 * A public read-only list of all the '<em><b>Action</b></em>' enumerators.
 * <!-- begin-user-doc -->
```

```
	 * <!-- end-user-doc -->
	 * @generated
	 */
	public static final List<Action> VALUES =
Collections.unmodifiableList(Arrays.asList(VALUES_ARRAY));

	/**
	 * Returns the '<em><b>Action</b></em>' literal with the specified literal value.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @generated
	 */
	public static Action get(String literal) {
	}

	/**
	 * Returns the '<em><b>Action</b></em>' literal with the specified name.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @generated
	 */
	public static Action getByName(String name) {
	}

	/**
	 * Returns the '<em><b>Action</b></em>' literal with the specified integer value.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @generated
	 */
	public static Action get(int value) {
	}

	/**
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @generated
	 */
	private final int value;

	/**
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @generated
	 */
	private final String name;

	/**
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @generated
	 */
	private final String literal;

	/**
	 * Only this class can construct instances.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @generated
	 */
	private Action(int value, String name, String literal) {
	}

	/**
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @generated
	 */
	public int getValue() {
	}

	/**
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @generated
	 */
	public String getName() {
	}

	/**
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @generated
	 */
	public String getLiteral() {
	}

	/**
	 * Returns the literal value of the enumerator, which is its string representation.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @generated
	 */
	@Override
	public String toString() {
	}

} //Action

/**
 * <copyright>
 * </copyright>
 *
 * $Id$
 */
package gcml;

import java.util.Arrays;
import java.util.Collections;
import java.util.List;

import org.eclipse.emf.common.util.Enumerator;

/**
 * <!-- begin-user-doc -->
 * A representation of the literals of the enumeration '<em><b>Capability</b></em>',
 * and utility methods for working with them.
 * <!-- end-user-doc -->
 * @see gcml.GcmlPackage#getCapability()
 * @model
 * @generated
 */
public enum Capability implements Enumerator {
	/**
	 * The '<em><b>Text File</b></em>' literal object.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @see #TEXT_FILE_VALUE
	 * @generated
	 * @ordered
	 */
	TEXT_FILE(0, "TextFile", "TextFile"),

	/**
	 * The '<em><b>Binary File</b></em>' literal object.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @see #BINARY_FILE_VALUE
	 * @generated
	 * @ordered
	 */
	BINARY_FILE(1, "BinaryFile", "BinaryFile"),

	/**
	 * The '<em><b>Stream File</b></em>' literal object.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @see #STREAM_FILE_VALUE
	 * @generated
	 * @ordered
	 */
	STREAM_FILE(2, "StreamFile", "StreamFile"),

	/**
	 * The '<em><b>Non Stream File</b></em>' literal object.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @see #NON_STREAM_FILE_VALUE
	 * @generated
	 * @ordered
	 */
	NON_STREAM_FILE(3, "NonStreamFile", "NonStreamFile"),

	/**
	 * The '<em><b>Audio File</b></em>' literal object.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @see #AUDIO_FILE_VALUE
	 * @generated
	 * @ordered
	 */
	AUDIO_FILE(4, "AudioFile", "AudioFile"),

	/**
	 * The '<em><b>Video File</b></em>' literal object.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
 * @see #VIDEO_FILE_VALUE
 * @generated
 * @ordered
 */
VIDEO_FILE(5, "VideoFile", "VideoFile"),

/**
 * The '<em><b>AV File</b></em>' literal object.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @see #AV_FILE_VALUE
 * @generated
 * @ordered
 */
AV_FILE(6, "AVFile", "AVFile"),

/**
 * The '<em><b>Text</b></em>' literal object.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @see #TEXT_VALUE
 * @generated
 * @ordered
 */
TEXT(7, "Text", "Text"),

/**
 * The '<em><b>Live Stream</b></em>' literal object.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @see #LIVE_STREAM_VALUE
 * @generated
 * @ordered
 */
LIVE_STREAM(8, "LiveStream", "LiveStream"),

/**
 * The '<em><b>Live Audio</b></em>' literal object.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @see #LIVE_AUDIO_VALUE
 * @generated
 * @ordered
 */
LIVE_AUDIO(9, "LiveAudio", "LiveAudio"),

/**
 * The '<em><b>Live Video</b></em>' literal object.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @see #LIVE_VIDEO_VALUE
 * @generated
 * @ordered
 */
LIVE_VIDEO(10, "LiveVideo", "LiveVideo"),

/**
 * The '<em><b>Live AV</b></em>' literal object.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @see #LIVE_AV_VALUE
 * @generated
 * @ordered
 */
LIVE_AV(11, "LiveAV", "LiveAV");

/**
 * The '<em><b>Text File</b></em>' literal value.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of '<em><b>Text File</b></em>' literal object isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @see #TEXT_FILE
 * @model name="TextFile"
 * @generated
 * @ordered
 */
public static final int TEXT_FILE_VALUE = 0;

/**
 * The '<em><b>Binary File</b></em>' literal value.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of '<em><b>Binary File</b></em>' literal object isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @see #BINARY_FILE
 * @model name="BinaryFile"
 * @generated
 * @ordered
 */
public static final int BINARY_FILE_VALUE = 1;

/**
 * The '<em><b>Stream File</b></em>' literal value.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of '<em><b>Stream File</b></em>' literal object isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @see #STREAM_FILE
 * @model name="StreamFile"
 * @generated
 * @ordered
 */
public static final int STREAM_FILE_VALUE = 2;

/**
 * The '<em><b>Non Stream File</b></em>' literal value.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of '<em><b>Non Stream File</b></em>' literal object isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @see #NON_STREAM_FILE
 * @model name="NonStreamFile"
 * @generated
 * @ordered
 */
public static final int NON_STREAM_FILE_VALUE = 3;

/**
 * The '<em><b>Audio File</b></em>' literal value.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of '<em><b>Audio File</b></em>' literal object isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @see #AUDIO_FILE
 * @model name="AudioFile"
 * @generated
 * @ordered
 */
public static final int AUDIO_FILE_VALUE = 4;

/**
 * The '<em><b>Video File</b></em>' literal value.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of '<em><b>Video File</b></em>' literal object isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @see #VIDEO_FILE
 * @model name="VideoFile"
 * @generated
 * @ordered
 */
public static final int VIDEO_FILE_VALUE = 5;

/**
 * The '<em><b>AV File</b></em>' literal value.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of '<em><b>AV File</b></em>' literal object isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @see #AV_FILE
 * @model name="AVFile"
 * @generated
 * @ordered
 */
public static final int AV_FILE_VALUE = 6;

/**
 * The '<em><b>Text</b></em>' literal value.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of '<em><b>Text</b></em>' literal object isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @see #TEXT
 * @model name="Text"
 * @generated
 * @ordered
 */
public static final int TEXT_VALUE = 7;

/**
 * The '<em><b>Live Stream</b></em>' literal value.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of '<em><b>Live Stream</b></em>' literal object isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @see #LIVE_STREAM
 * @model name="LiveStream"
 * @generated
 * @ordered
 */
public static final int LIVE_STREAM_VALUE = 8;

/**
 * The '<em><b>Live Audio</b></em>' literal value.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of '<em><b>Live Audio</b></em>' literal object isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @see #LIVE_AUDIO
 * @model name="LiveAudio"
 * @generated
 * @ordered
 */
public static final int LIVE_AUDIO_VALUE = 9;

/**
 * The '<em><b>Live Video</b></em>' literal value.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of '<em><b>Live Video</b></em>' literal object isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @see #LIVE_VIDEO
 * @model name="LiveVideo"
 * @generated
 * @ordered
 */
public static final int LIVE_VIDEO_VALUE = 10;

/**
 * The '<em><b>Live AV</b></em>' literal value.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of '<em><b>Live AV</b></em>' literal object isn't clear,
         * there really should be more of a description here...
         * </p>
         * <!-- end-user-doc -->
         * @see #LIVE_AV
         * @model name="LiveAV"
         * @generated
         * @ordered
         */
        public static final int LIVE_AV_VALUE = 11;

        /**
         * An array of all the '<em><b>Capability</b></em>' enumerators.
         * <!-- begin-user-doc -->
         * <!-- end-user-doc -->
         * @generated
         */
        private static final Capability[] VALUES_ARRAY =
                new Capability[] {
                        TEXT_FILE,
                        BINARY_FILE,
                        STREAM_FILE,
                        NON_STREAM_FILE,
                        AUDIO_FILE,
                        VIDEO_FILE,
                        AV_FILE,
                        TEXT,
                        LIVE_STREAM,
                        LIVE_AUDIO,
                        LIVE_VIDEO,
                        LIVE_AV,
                };

        /**
         * A public read-only list of all the '<em><b>Capability</b></em>' enumerators.
         * <!-- begin-user-doc -->
         * <!-- end-user-doc -->
         * @generated
         */
        public          static          final          List<Capability>          VALUES          =
Collections.unmodifiableList(Arrays.asList(VALUES_ARRAY));

        /**
         * Returns the '<em><b>Capability</b></em>' literal with the specified literal value.
         * <!-- begin-user-doc -->
         * <!-- end-user-doc -->
         * @generated
         */
        public static Capability get(String literal) {
        }

        /**
         * Returns the '<em><b>Capability</b></em>' literal with the specified name.
         * <!-- begin-user-doc -->
         * <!-- end-user-doc -->
         * @generated
         */
        public static Capability getByName(String name) {
        }
```

```
/**
 * Returns the '<em><b>Capability</b></em>' literal with the specified integer value.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
public static Capability get(int value) {
}

/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
private final int value;

/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
private final String name;

/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
private final String literal;

/**
 * Only this class can construct instances.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
private Capability(int value, String name, String literal) {
}

/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
public int getValue() {
}

/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
public String getName() {
}

/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
```

```
         */
        public String getLiteral() {
        }

        /**
         * Returns the literal value of the enumerator, which is its string representation.
         * <!-- begin-user-doc -->
         * <!-- end-user-doc -->
         * @generated
         */
        @Override
        public String toString() {
        }

} //Capability

/**
 * <copyright>
 * </copyright>
 *
 * $Id$
 */
package gcml;


/**
 * <!-- begin-user-doc -->
 * A representation of the model object '<em><b>Child Form</b></em>'.
 * <!-- end-user-doc -->
 *
 * <p>
 * The following features are supported:
 * <ul>
 *   <li>{@link gcml.ChildForm#getToParentForm <em>To Parent Form</em>}</li>
 * </ul>
 * </p>
 *
 * @see gcml.GcmlPackage#getChildForm()
 * @model
 * @generated
 */
public interface ChildForm extends Form {
        /**
         * Returns the value of the '<em><b>To Parent Form</b></em>' reference.
         * <!-- begin-user-doc -->
         * <p>
         * If the meaning of the '<em>To Parent Form</em>' reference isn't clear,
         * there really should be more of a description here...
         * </p>
         * <!-- end-user-doc -->
         * @return the value of the '<em>To Parent Form</em>' reference.
         * @see #setToParentForm(Form)
         * @see gcml.GcmlPackage#getChildForm_ToParentForm()
         * @model required="true"
         * @generated
         */
        Form getToParentForm();

        /**
```

```
	 * Sets the value of the '{@link gcml.ChildForm#getToParentForm <em>To Parent Form</em>}' 
reference.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @param value the new value of the '<em>To Parent Form</em>' reference.
	 * @see #getToParentForm()
	 * @generated
	 */
	void setToParentForm(Form value);

} // ChildForm

/**
 * <copyright>
 * </copyright>
 *
 * $Id$
 */
package gcml;


/**
 * <!-- begin-user-doc -->
 * A representation of the model object '<em><b>Child Medium</b></em>'.
 * <!-- end-user-doc -->
 *
 * <p>
 * The following features are supported:
 * <ul>
 *   <li>{@link gcml.ChildMedium#getToParentForm <em>To Parent Form</em>}</li>
 * </ul>
 * </p>
 *
 * @see gcml.GcmlPackage#getChildMedium()
 * @model
 * @generated
 */
public interface ChildMedium extends Medium {
	/**
	 * Returns the value of the '<em><b>To Parent Form</b></em>' reference.
	 * <!-- begin-user-doc -->
	 * <p>
	 * If the meaning of the '<em>To Parent Form</em>' reference isn't clear,
	 * there really should be more of a description here...
	 * </p>
	 * <!-- end-user-doc -->
	 * @return the value of the '<em>To Parent Form</em>' reference.
	 * @see #setToParentForm(Form)
	 * @see gcml.GcmlPackage#getChildMedium_ToParentForm()
	 * @model required="true"
	 * @generated
	 */
	Form getToParentForm();

	/**
	 * Sets the value of the '{@link gcml.ChildMedium#getToParentForm <em>To Parent Form</em>}' 
reference.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @param value the new value of the '<em>To Parent Form</em>' reference.
	 * @see #getToParentForm()
	 * @generated
	 */
	void setToParentForm(Form value);

} // ChildMedium


/**
 * <copyright>
 * </copyright>
 *
 * $Id$
 */
package gcml;

import org.eclipse.emf.ecore.EObject;

/**
 * <!-- begin-user-doc -->
 * A representation of the model object '<em><b>Connection</b></em>'.
 * <!-- end-user-doc -->
 *
 * <p>
 * The following features are supported:
 * <ul>
 *   <li>{@link gcml.Connection#getBandwidth <em>Bandwidth</em>}</li>
 *   <li>{@link gcml.Connection#getConnectionID <em>Connection ID</em>}</li>
 * </ul>
 * </p>
 *
 * @see gcml.GcmlPackage#getConnection()
 * @model
 * @generated
 */
public interface Connection extends EObject {
	/**
	 * Returns the value of the '<em><b>Bandwidth</b></em>' attribute.
	 * <!-- begin-user-doc -->
	 * <p>
	 * If the meaning of the '<em>Bandwidth</em>' attribute isn't clear,
	 * there really should be more of a description here...
	 * </p>
	 * <!-- end-user-doc -->
	 * @return the value of the '<em>Bandwidth</em>' attribute.
	 * @see #setBandwidth(String)
	 * @see gcml.GcmlPackage#getConnection_Bandwidth()
	 * @model dataType="org.eclipse.emf.ecore.xml.type.String"
	 * @generated
	 */
	String getBandwidth();

	/**
	 * Sets the value of the '{@link gcml.Connection#getBandwidth <em>Bandwidth</em>}' attribute.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @param value the new value of the '<em>Bandwidth</em>' attribute.
	 * @see #getBandwidth()
	 * @generated
	 */
	void setBandwidth(String value);

	/**
	 * Returns the value of the '<em><b>Connection ID</b></em>' attribute.
	 * <!-- begin-user-doc -->
	 * <p>
	 * If the meaning of the '<em>Connection ID</em>' attribute isn't clear,
	 * there really should be more of a description here...
	 * </p>
	 * <!-- end-user-doc -->
	 * @return the value of the '<em>Connection ID</em>' attribute.
	 * @see #setConnectionID(String)
	 * @see gcml.GcmlPackage#getConnection_ConnectionID()
	 * @model dataType="org.eclipse.emf.ecore.xml.type.String" required="true"
	 * @generated
	 */
	String getConnectionID();

	/**
	 * Sets the value of the '{@link gcml.Connection#getConnectionID <em>Connection ID</em>}' attribute.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @param value the new value of the '<em>Connection ID</em>' attribute.
	 * @see #getConnectionID()
	 * @generated
	 */
	void setConnectionID(String value);

} // Connection


/**
 * <copyright>
 * </copyright>
 *
 * $Id$
 */
package gcml;

import org.eclipse.emf.common.util.EList;

import org.eclipse.emf.ecore.EObject;

/**
 * <!-- begin-user-doc -->
 * A representation of the model object '<em><b>Device</b></em>'.
 * <!-- end-user-doc -->
 *
 * <p>
 * The following features are supported:
 * <ul>
 *   <li>{@link gcml.Device#getDeviceCapability <em>Device Capability</em>}</li>
 *   <li>{@link gcml.Device#getDeviceID <em>Device ID</em>}</li>
 *   <li>{@link gcml.Device#isIsLocal <em>Is Local</em>}</li>
 *   <li>{@link gcml.Device#isIsVirtual <em>Is Virtual</em>}</li>
 *   <li>{@link gcml.Device#getToConnection <em>To Connection</em>}</li>
 * </ul>
 * </p>
 *
 * @see gcml.GcmlPackage#getDevice()
 * @model
 * @generated
 */
public interface Device extends EObject {
	/**
	 * Returns the value of the '<em><b>Device Capability</b></em>' attribute list.
	 * The list contents are of type {@link gcml.Capability}.
	 * The literals are from the enumeration {@link gcml.Capability}.
	 * <!-- begin-user-doc -->
	 * <p>
	 * If the meaning of the '<em>Device Capability</em>' attribute list isn't clear,
	 * there really should be more of a description here...
	 * </p>
	 * <!-- end-user-doc -->
	 * @return the value of the '<em>Device Capability</em>' attribute list.
	 * @see gcml.Capability
	 * @see gcml.GcmlPackage#getDevice_DeviceCapability()
	 * @model unique="false"
	 * @generated
	 */
	EList<Capability> getDeviceCapability();

	/**
	 * Returns the value of the '<em><b>Device ID</b></em>' attribute.
	 * <!-- begin-user-doc -->
	 * <p>
	 * If the meaning of the '<em>Device ID</em>' attribute isn't clear,
	 * there really should be more of a description here...
	 * </p>
	 * <!-- end-user-doc -->
	 * @return the value of the '<em>Device ID</em>' attribute.
	 * @see #setDeviceID(String)
	 * @see gcml.GcmlPackage#getDevice_DeviceID()
	 * @model dataType="org.eclipse.emf.ecore.xml.type.String" required="true"
	 * @generated
	 */
	String getDeviceID();

	/**
	 * Sets the value of the '{@link gcml.Device#getDeviceID <em>Device ID</em>}' attribute.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @param value the new value of the '<em>Device ID</em>' attribute.
	 * @see #getDeviceID()
	 * @generated
	 */
	void setDeviceID(String value);

	/**
	 * Returns the value of the '<em><b>Is Local</b></em>' attribute.
	 * <!-- begin-user-doc -->
	 * <p>
	 * If the meaning of the '<em>Is Local</em>' attribute isn't clear,
	 * there really should be more of a description here...
	 * </p>
	 * <!-- end-user-doc -->
 * @return the value of the '<em>Is Local</em>' attribute.
 * @see #isSetIsLocal()
 * @see #unsetIsLocal()
 * @see #setIsLocal(boolean)
 * @see gcml.GcmlPackage#getDevice_IsLocal()
 * @model unsettable="true" dataType="org.eclipse.emf.ecore.xml.type.Boolean"
 * @generated
 */
boolean isIsLocal();

/**
 * Sets the value of the '{@link gcml.Device#isIsLocal <em>Is Local</em>}' attribute.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @param value the new value of the '<em>Is Local</em>' attribute.
 * @see #isSetIsLocal()
 * @see #unsetIsLocal()
 * @see #isIsLocal()
 * @generated
 */
void setIsLocal(boolean value);

/**
 * Unsets the value of the '{@link gcml.Device#isIsLocal <em>Is Local</em>}' attribute.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @see #isSetIsLocal()
 * @see #isIsLocal()
 * @see #setIsLocal(boolean)
 * @generated
 */
void unsetIsLocal();

/**
 * Returns whether the value of the '{@link gcml.Device#isIsLocal <em>Is Local</em>}' attribute is set.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @return whether the value of the '<em>Is Local</em>' attribute is set.
 * @see #unsetIsLocal()
 * @see #isIsLocal()
 * @see #setIsLocal(boolean)
 * @generated
 */
boolean isSetIsLocal();

/**
 * Returns the value of the '<em><b>Is Virtual</b></em>' attribute.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of the '<em>Is Virtual</em>' attribute isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @return the value of the '<em>Is Virtual</em>' attribute.
 * @see #isSetIsVirtual()
 * @see #unsetIsVirtual()
 * @see #setIsVirtual(boolean)
 * @see gcml.GcmlPackage#getDevice_IsVirtual()
 * @model unsettable="true" dataType="org.eclipse.emf.ecore.xml.type.Boolean"
 * @generated
 */
boolean isIsVirtual();

/**
 * Sets the value of the '{@link gcml.Device#isIsVirtual <em>Is Virtual</em>}' attribute.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @param value the new value of the '<em>Is Virtual</em>' attribute.
 * @see #isSetIsVirtual()
 * @see #unsetIsVirtual()
 * @see #isIsVirtual()
 * @generated
 */
void setIsVirtual(boolean value);

/**
 * Unsets the value of the '{@link gcml.Device#isIsVirtual <em>Is Virtual</em>}' attribute.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @see #isSetIsVirtual()
 * @see #isIsVirtual()
 * @see #setIsVirtual(boolean)
 * @generated
 */
void unsetIsVirtual();

/**
 * Returns whether the value of the '{@link gcml.Device#isIsVirtual <em>Is Virtual</em>}' attribute is
set.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @return whether the value of the '<em>Is Virtual</em>' attribute is set.
 * @see #unsetIsVirtual()
 * @see #isIsVirtual()
 * @see #setIsVirtual(boolean)
 * @generated
 */
boolean isSetIsVirtual();

/**
 * Returns the value of the '<em><b>To Connection</b></em>' reference.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of the '<em>To Connection</em>' reference isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @return the value of the '<em>To Connection</em>' reference.
 * @see #setToConnection(Connection)
 * @see gcml.GcmlPackage#getDevice_ToConnection()
 * @model required="true"
 * @generated
 */
Connection getToConnection();

/**
 * Sets the value of the '{@link gcml.Device#getToConnection <em>To Connection</em>}' reference.
 * <!-- begin-user-doc -->
```

```
		 * <!-- end-user-doc -->
		 * @param value the new value of the '<em>To Connection</em>' reference.
		 * @see #getToConnection()
		 * @generated
		 */
		void setToConnection(Connection value);

} // Device

/**
 * <copyright>
 * </copyright>
 *
 * $Id$
 */
package gcml;

import org.eclipse.emf.common.util.EList;

import org.eclipse.emf.ecore.EObject;

/**
 * <!-- begin-user-doc -->
 * A representation of the model object '<em><b>Form</b></em>'.
 * <!-- end-user-doc -->
 *
 * <p>
 * The following features are supported:
 * <ul>
 *   <li>{@link gcml.Form#getMediumDataType <em>Medium Data Type</em>}</li>
 *   <li>{@link gcml.Form#getAction <em>Action</em>}</li>
 *   <li>{@link gcml.Form#getFormName <em>Form Name</em>}</li>
 *   <li>{@link gcml.Form#getSuggestedApplication <em>Suggested Application</em>}</li>
 *   <li>{@link gcml.Form#getVoiceCommand <em>Voice Command</em>}</li>
 * </ul>
 * </p>
 *
 * @see gcml.GcmlPackage#getForm()
 * @model
 * @generated
 */
public interface Form extends EObject {
		/**
		 * Returns the value of the '<em><b>Medium Data Type</b></em>' attribute list.
		 * The list contents are of type {@link java.lang.String}.
		 * <!-- begin-user-doc -->
		 * <p>
		 * If the meaning of the '<em>Medium Data Type</em>' attribute list isn't clear,
		 * there really should be more of a description here...
		 * </p>
		 * <!-- end-user-doc -->
		 * @return the value of the '<em>Medium Data Type</em>' attribute list.
		 * @see gcml.GcmlPackage#getForm_MediumDataType()
		 * @model unique="false" dataType="org.eclipse.emf.ecore.xml.type.String"
		 * @generated
		 */
		EList<String> getMediumDataType();

		/**
 * Returns the value of the '<em><b>Action</b></em>' attribute.
 * The literals are from the enumeration {@link gcml.Action}.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of the '<em>Action</em>' attribute isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @return the value of the '<em>Action</em>' attribute.
 * @see gcml.Action
 * @see #isSetAction()
 * @see #unsetAction()
 * @see #setAction(Action)
 * @see gcml.GcmlPackage#getForm_Action()
 * @model unsettable="true"
 * @generated
 */
Action getAction();

/**
 * Sets the value of the '{@link gcml.Form#getAction <em>Action</em>}' attribute.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @param value the new value of the '<em>Action</em>' attribute.
 * @see gcml.Action
 * @see #isSetAction()
 * @see #unsetAction()
 * @see #getAction()
 * @generated
 */
void setAction(Action value);

/**
 * Unsets the value of the '{@link gcml.Form#getAction <em>Action</em>}' attribute.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @see #isSetAction()
 * @see #getAction()
 * @see #setAction(Action)
 * @generated
 */
void unsetAction();

/**
 * Returns whether the value of the '{@link gcml.Form#getAction <em>Action</em>}' attribute is set.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @return whether the value of the '<em>Action</em>' attribute is set.
 * @see #unsetAction()
 * @see #getAction()
 * @see #setAction(Action)
 * @generated
 */
boolean isSetAction();

/**
 * Returns the value of the '<em><b>Form Name</b></em>' attribute.
 * <!-- begin-user-doc -->
 * <p>
```

```
         * If the meaning of the '<em>Form Name</em>' attribute isn't clear,
         * there really should be more of a description here...
         * </p>
         * <!-- end-user-doc -->
         * @return the value of the '<em>Form Name</em>' attribute.
         * @see #setFormName(String)
         * @see gcml.GcmlPackage#getForm_FormName()
         * @model dataType="org.eclipse.emf.ecore.xml.type.String" required="true"
         * @generated
         */
        String getFormName();

        /**
         * Sets the value of the '{@link gcml.Form#getFormName <em>Form Name</em>}' attribute.
         * <!-- begin-user-doc -->
         * <!-- end-user-doc -->
         * @param value the new value of the '<em>Form Name</em>' attribute.
         * @see #getFormName()
         * @generated
         */
        void setFormName(String value);

        /**
         * Returns the value of the '<em><b>Suggested Application</b></em>' attribute.
         * <!-- begin-user-doc -->
         * <p>
         * If the meaning of the '<em>Suggested Application</em>' attribute isn't clear,
         * there really should be more of a description here...
         * </p>
         * <!-- end-user-doc -->
         * @return the value of the '<em>Suggested Application</em>' attribute.
         * @see #setSuggestedApplication(String)
         * @see gcml.GcmlPackage#getForm_SuggestedApplication()
         * @model dataType="org.eclipse.emf.ecore.xml.type.String"
         * @generated
         */
        String getSuggestedApplication();

        /**
         * Sets the value of the '{@link gcml.Form#getSuggestedApplication <em>Suggested Application</em>}'
attribute.
         * <!-- begin-user-doc -->
         * <!-- end-user-doc -->
         * @param value the new value of the '<em>Suggested Application</em>' attribute.
         * @see #getSuggestedApplication()
         * @generated
         */
        void setSuggestedApplication(String value);

        /**
         * Returns the value of the '<em><b>Voice Command</b></em>' attribute.
         * <!-- begin-user-doc -->
         * <p>
         * If the meaning of the '<em>Voice Command</em>' attribute isn't clear,
         * there really should be more of a description here...
         * </p>
         * <!-- end-user-doc -->
         * @return the value of the '<em>Voice Command</em>' attribute.
         * @see #setVoiceCommand(String)
```

```
	 * @see gcml.GcmlPackage#getForm_VoiceCommand()
	 * @model dataType="org.eclipse.emf.ecore.xml.type.String"
	 * @generated
	 */
	String getVoiceCommand();

	/**
	 * Sets the value of the '{@link gcml.Form#getVoiceCommand <em>Voice Command</em>}' attribute.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @param value the new value of the '<em>Voice Command</em>' attribute.
	 * @see #getVoiceCommand()
	 * @generated
	 */
	void setVoiceCommand(String value);

} // Form


/**
 * <copyright>
 * </copyright>
 *
 * $Id$
 */
package gcml;

import org.eclipse.emf.common.util.EList;

import org.eclipse.emf.ecore.EObject;

/**
 * <!-- begin-user-doc -->
 * A representation of the model object '<em><b>Gcml</b></em>'.
 * <!-- end-user-doc -->
 *
 * <p>
 * The following features are supported:
 * <ul>
 *   <li>{@link gcml.Gcml#getConnection <em>Connection</em>}</li>
 *   <li>{@link gcml.Gcml#getMedium <em>Medium</em>}</li>
 *   <li>{@link gcml.Gcml#getForm <em>Form</em>}</li>
 *   <li>{@link gcml.Gcml#getPerson <em>Person</em>}</li>
 *   <li>{@link gcml.Gcml#getIsAttached <em>Is Attached</em>}</li>
 *   <li>{@link gcml.Gcml#getDevice <em>Device</em>}</li>
 *   <li>{@link gcml.Gcml#getChildMedium <em>Child Medium</em>}</li>
 *   <li>{@link gcml.Gcml#getChildForm <em>Child Form</em>}</li>
 * </ul>
 * </p>
 *
 * @see gcml.GcmlPackage#getGcml()
 * @model
 * @generated
 */
public interface Gcml extends EObject {
	/**
	 * Returns the value of the '<em><b>Connection</b></em>' containment reference list.
	 * The list contents are of type {@link gcml.Connection}.
	 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of the '<em>Connection</em>' containment reference list isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @return the value of the '<em>Connection</em>' containment reference list.
 * @see gcml.GcmlPackage#getGcml_Connection()
 * @model containment="true"
 * @generated
 */
EList<Connection> getConnection();

/**
 * Returns the value of the '<em><b>Medium</b></em>' containment reference list.
 * The list contents are of type {@link gcml.MainMedium}.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of the '<em>Medium</em>' containment reference list isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @return the value of the '<em>Medium</em>' containment reference list.
 * @see gcml.GcmlPackage#getGcml_Medium()
 * @model containment="true"
 * @generated
 */
EList<MainMedium> getMedium();

/**
 * Returns the value of the '<em><b>Form</b></em>' containment reference list.
 * The list contents are of type {@link gcml.MainForm}.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of the '<em>Form</em>' containment reference list isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @return the value of the '<em>Form</em>' containment reference list.
 * @see gcml.GcmlPackage#getGcml_Form()
 * @model containment="true"
 * @generated
 */
EList<MainForm> getForm();

/**
 * Returns the value of the '<em><b>Person</b></em>' containment reference list.
 * The list contents are of type {@link gcml.Person}.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of the '<em>Person</em>' containment reference list isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @return the value of the '<em>Person</em>' containment reference list.
 * @see gcml.GcmlPackage#getGcml_Person()
 * @model containment="true"
 * @generated
 */
EList<Person> getPerson();
```

```
/**
 * Returns the value of the '<em><b>Is Attached</b></em>' containment reference list.
 * The list contents are of type {@link gcml.IsAttached}.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of the '<em>Is Attached</em>' containment reference list isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @return the value of the '<em>Is Attached</em>' containment reference list.
 * @see gcml.GcmlPackage#getGcml_IsAttached()
 * @model containment="true"
 * @generated
 */
EList<IsAttached> getIsAttached();

/**
 * Returns the value of the '<em><b>Device</b></em>' containment reference list.
 * The list contents are of type {@link gcml.Device}.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of the '<em>Device</em>' containment reference list isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @return the value of the '<em>Device</em>' containment reference list.
 * @see gcml.GcmlPackage#getGcml_Device()
 * @model containment="true"
 * @generated
 */
EList<Device> getDevice();

/**
 * Returns the value of the '<em><b>Child Medium</b></em>' reference list.
 * The list contents are of type {@link gcml.ChildMedium}.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of the '<em>Child Medium</em>' reference list isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @return the value of the '<em>Child Medium</em>' reference list.
 * @see gcml.GcmlPackage#getGcml_ChildMedium()
 * @model
 * @generated
 */
EList<ChildMedium> getChildMedium();

/**
 * Returns the value of the '<em><b>Child Form</b></em>' reference list.
 * The list contents are of type {@link gcml.ChildForm}.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of the '<em>Child Form</em>' reference list isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @return the value of the '<em>Child Form</em>' reference list.
```

```
	 * @see gcml.GcmlPackage#getGcml_ChildForm()
	 * @model
	 * @generated
	 */
	EList<ChildForm> getChildForm();

} // Gcml


/**
 * <copyright>
 * </copyright>
 *
 * $Id$
 */
package gcml;

import org.eclipse.emf.ecore.EObject;

/**
 * <!-- begin-user-doc -->
 * A representation of the model object '<em><b>Is Attached</b></em>'.
 * <!-- end-user-doc -->
 *
 * <p>
 * The following features are supported:
 * <ul>
 *   <li>{@link gcml.IsAttached#getToDevice <em>To Device</em>}</li>
 * </ul>
 * </p>
 *
 * @see gcml.GcmlPackage#getIsAttached()
 * @model
 * @generated
 */
public interface IsAttached extends EObject {
	/**
	 * Returns the value of the '<em><b>To Device</b></em>' containment reference.
	 * <!-- begin-user-doc -->
	 * <p>
	 * If the meaning of the '<em>To Device</em>' containment reference isn't clear,
	 * there really should be more of a description here...
	 * </p>
	 * <!-- end-user-doc -->
	 * @return the value of the '<em>To Device</em>' containment reference.
	 * @see #setToDevice(Device)
	 * @see gcml.GcmlPackage#getIsAttached_ToDevice()
	 * @model containment="true" required="true"
	 * @generated
	 */
	Device getToDevice();

	/**
	 * Sets the value of the '{@link gcml.IsAttached#getToDevice <em>To Device</em>}' containment
reference.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @param value the new value of the '<em>To Device</em>' containment reference.
	 * @see #getToDevice()
```

```
		 * @generated
		 */
		void setToDevice(Device value);

} // IsAttached

/**
 * <copyright>
 * </copyright>
 *
 * $Id$
 */
package gcml;


/**
 * <!-- begin-user-doc -->
 * A representation of the model object '<em><b>Main Form</b></em>'.
 * <!-- end-user-doc -->
 *
 * <p>
 * The following features are supported:
 * <ul>
 *   <li>{@link gcml.MainForm#getToConnection <em>To Connection</em>}</li>
 * </ul>
 * </p>
 *
 * @see gcml.GcmlPackage#getMainForm()
 * @model
 * @generated
 */
public interface MainForm extends Form {
		/**
		 * Returns the value of the '<em><b>To Connection</b></em>' reference.
		 * <!-- begin-user-doc -->
		 * <p>
		 * If the meaning of the '<em>To Connection</em>' reference isn't clear,
		 * there really should be more of a description here...
		 * </p>
		 * <!-- end-user-doc -->
		 * @return the value of the '<em>To Connection</em>' reference.
		 * @see #setToConnection(Connection)
		 * @see gcml.GcmlPackage#getMainForm_ToConnection()
		 * @model
		 * @generated
		 */
		Connection getToConnection();

		/**
		 * Sets the value of the '{@link gcml.MainForm#getToConnection <em>To Connection</em>}' reference.
		 * <!-- begin-user-doc -->
		 * <!-- end-user-doc -->
		 * @param value the new value of the '<em>To Connection</em>' reference.
		 * @see #getToConnection()
		 * @generated
		 */
		void setToConnection(Connection value);

} // MainForm
```

```
/**
 * <copyright>
 * </copyright>
 *
 * $Id$
 */
package gcml;


/**
 * <!-- begin-user-doc -->
 * A representation of the model object '<em><b>Main Medium</b></em>'.
 * <!-- end-user-doc -->
 *
 * <p>
 * The following features are supported:
 * <ul>
 *   <li>{@link gcml.MainMedium#getToConnection <em>To Connection</em>}</li>
 * </ul>
 * </p>
 *
 * @see gcml.GcmlPackage#getMainMedium()
 * @model
 * @generated
 */
public interface MainMedium extends Medium {
	/**
	 * Returns the value of the '<em><b>To Connection</b></em>' reference.
	 * <!-- begin-user-doc -->
	 * <p>
	 * If the meaning of the '<em>To Connection</em>' reference isn't clear,
	 * there really should be more of a description here...
	 * </p>
	 * <!-- end-user-doc -->
	 * @return the value of the '<em>To Connection</em>' reference.
	 * @see #setToConnection(Connection)
	 * @see gcml.GcmlPackage#getMainMedium_ToConnection()
	 * @model required="true"
	 * @generated
	 */
	Connection getToConnection();

	/**
	 * Sets the value of the '{@link gcml.MainMedium#getToConnection <em>To Connection</em>}' 
reference.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @param value the new value of the '<em>To Connection</em>' reference.
	 * @see #getToConnection()
	 * @generated
	 */
	void setToConnection(Connection value);

} // MainMedium

/**
 * <copyright>
 * </copyright>
 *
 * $Id$
 */
package gcml;

import org.eclipse.emf.ecore.EObject;

/**
 * <!-- begin-user-doc -->
 * A representation of the model object '<em><b>Medium</b></em>'.
 * <!-- end-user-doc -->
 *
 * <p>
 * The following features are supported:
 * <ul>
 *   <li>{@link gcml.Medium#getDerivedFromBuiltInType <em>Derived From Built In Type</em>}</li>
 *   <li>{@link gcml.Medium#getMediumName <em>Medium Name</em>}</li>
 *   <li>{@link gcml.Medium#getSuggestedApplication <em>Suggested Application</em>}</li>
 *   <li>{@link gcml.Medium#getVoiceCommand <em>Voice Command</em>}</li>
 * </ul>
 * </p>
 *
 * @see gcml.GcmlPackage#getMedium()
 * @model
 * @generated
 */
public interface Medium extends EObject {
		/**
		 * Returns the value of the '<em><b>Derived From Built In Type</b></em>' attribute.
		 * The literals are from the enumeration {@link gcml.Capability}.
		 * <!-- begin-user-doc -->
		 * <p>
		 * If the meaning of the '<em>Derived From Built In Type</em>' attribute isn't clear,
		 * there really should be more of a description here...
		 * </p>
		 * <!-- end-user-doc -->
		 * @return the value of the '<em>Derived From Built In Type</em>' attribute.
		 * @see gcml.Capability
		 * @see #isSetDerivedFromBuiltInType()
		 * @see #unsetDerivedFromBuiltInType()
		 * @see #setDerivedFromBuiltInType(Capability)
		 * @see gcml.GcmlPackage#getMedium_DerivedFromBuiltInType()
		 * @model unsettable="true"
		 * @generated
		 */
		Capability getDerivedFromBuiltInType();

		/**
		 * Sets the value of the '{@link gcml.Medium#getDerivedFromBuiltInType <em>Derived From Built In
Type</em>}' attribute.
		 * <!-- begin-user-doc -->
		 * <!-- end-user-doc -->
		 * @param value the new value of the '<em>Derived From Built In Type</em>' attribute.
		 * @see gcml.Capability
		 * @see #isSetDerivedFromBuiltInType()
		 * @see #unsetDerivedFromBuiltInType()
		 * @see #getDerivedFromBuiltInType()
		 * @generated
		 */
	void setDerivedFromBuiltInType(Capability value);

	/**
	 * Unsets the value of the '{@link gcml.Medium#getDerivedFromBuiltInType <em>Derived From Built
In Type</em>}' attribute.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @see #isSetDerivedFromBuiltInType()
	 * @see #getDerivedFromBuiltInType()
	 * @see #setDerivedFromBuiltInType(Capability)
	 * @generated
	 */
	void unsetDerivedFromBuiltInType();

	/**
	 * Returns whether the value of the '{@link gcml.Medium#getDerivedFromBuiltInType <em>Derived
From Built In Type</em>}' attribute is set.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @return whether the value of the '<em>Derived From Built In Type</em>' attribute is set.
	 * @see #unsetDerivedFromBuiltInType()
	 * @see #getDerivedFromBuiltInType()
	 * @see #setDerivedFromBuiltInType(Capability)
	 * @generated
	 */
	boolean isSetDerivedFromBuiltInType();

	/**
	 * Returns the value of the '<em><b>Medium Name</b></em>' attribute.
	 * <!-- begin-user-doc -->
	 * <p>
	 * If the meaning of the '<em>Medium Name</em>' attribute isn't clear,
	 * there really should be more of a description here...
	 * </p>
	 * <!-- end-user-doc -->
	 * @return the value of the '<em>Medium Name</em>' attribute.
	 * @see #setMediumName(String)
	 * @see gcml.GcmlPackage#getMedium_MediumName()
	 * @model dataType="org.eclipse.emf.ecore.xml.type.String" required="true"
	 * @generated
	 */
	String getMediumName();

	/**
	 * Sets the value of the '{@link gcml.Medium#getMediumName <em>Medium Name</em>}' attribute.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @param value the new value of the '<em>Medium Name</em>' attribute.
	 * @see #getMediumName()
	 * @generated
	 */
	void setMediumName(String value);

	/**
	 * Returns the value of the '<em><b>Suggested Application</b></em>' attribute.
	 * <!-- begin-user-doc -->
	 * <p>
	 * If the meaning of the '<em>Suggested Application</em>' attribute isn't clear,
	 * there really should be more of a description here...
	 * </p>
	 * <!-- end-user-doc -->
	 * @return the value of the '<em>Suggested Application</em>' attribute.
	 * @see #setSuggestedApplication(String)
	 * @see gcml.GcmlPackage#getMedium_SuggestedApplication()
	 * @model dataType="org.eclipse.emf.ecore.xml.type.String"
	 * @generated
	 */
	String getSuggestedApplication();

	/**
	 * Sets the value of the '{@link gcml.Medium#getSuggestedApplication <em>Suggested
Application</em>}' attribute.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @param value the new value of the '<em>Suggested Application</em>' attribute.
	 * @see #getSuggestedApplication()
	 * @generated
	 */
	void setSuggestedApplication(String value);

	/**
	 * Returns the value of the '<em><b>Voice Command</b></em>' attribute.
	 * <!-- begin-user-doc -->
	 * <p>
	 * If the meaning of the '<em>Voice Command</em>' attribute isn't clear,
	 * there really should be more of a description here...
	 * </p>
	 * <!-- end-user-doc -->
	 * @return the value of the '<em>Voice Command</em>' attribute.
	 * @see #setVoiceCommand(String)
	 * @see gcml.GcmlPackage#getMedium_VoiceCommand()
	 * @model dataType="org.eclipse.emf.ecore.xml.type.String"
	 * @generated
	 */
	String getVoiceCommand();

	/**
	 * Sets the value of the '{@link gcml.Medium#getVoiceCommand <em>Voice Command</em>}'
attribute.
	 * <!-- begin-user-doc -->
	 * <!-- end-user-doc -->
	 * @param value the new value of the '<em>Voice Command</em>' attribute.
	 * @see #getVoiceCommand()
	 * @generated
	 */
	void setVoiceCommand(String value);

} // Medium

/**
 * <copyright>
 * </copyright>
 *
 * $Id$
 */
package gcml;

import org.eclipse.emf.ecore.EObject;
```

```java
/**
 * <!-- begin-user-doc -->
 * A representation of the model object '<em><b>Person</b></em>'.
 * <!-- end-user-doc -->
 *
 * <p>
 * The following features are supported:
 * <ul>
 *   <li>{@link gcml.Person#getPersonID <em>Person ID</em>}</li>
 *   <li>{@link gcml.Person#getPersonName <em>Person Name</em>}</li>
 *   <li>{@link gcml.Person#getPersonRole <em>Person Role</em>}</li>
 *   <li>{@link gcml.Person#getToIsAttached <em>To Is Attached</em>}</li>
 * </ul>
 * </p>
 *
 * @see gcml.GcmlPackage#getPerson()
 * @model
 * @generated
 */
public interface Person extends EObject {
		/**
		 * Returns the value of the '<em><b>Person ID</b></em>' attribute.
		 * <!-- begin-user-doc -->
		 * <p>
		 * If the meaning of the '<em>Person ID</em>' attribute isn't clear,
		 * there really should be more of a description here...
		 * </p>
		 * <!-- end-user-doc -->
		 * @return the value of the '<em>Person ID</em>' attribute.
		 * @see #setPersonID(String)
		 * @see gcml.GcmlPackage#getPerson_PersonID()
		 * @model dataType="org.eclipse.emf.ecore.xml.type.String" required="true"
		 * @generated
		 */
		String getPersonID();

		/**
		 * Sets the value of the '{@link gcml.Person#getPersonID <em>Person ID</em>}' attribute.
		 * <!-- begin-user-doc -->
		 * <!-- end-user-doc -->
		 * @param value the new value of the '<em>Person ID</em>' attribute.
		 * @see #getPersonID()
		 * @generated
		 */
		void setPersonID(String value);

		/**
		 * Returns the value of the '<em><b>Person Name</b></em>' attribute.
		 * <!-- begin-user-doc -->
		 * <p>
		 * If the meaning of the '<em>Person Name</em>' attribute isn't clear,
		 * there really should be more of a description here...
		 * </p>
		 * <!-- end-user-doc -->
		 * @return the value of the '<em>Person Name</em>' attribute.
		 * @see #setPersonName(String)
		 * @see gcml.GcmlPackage#getPerson_PersonName()
		 * @model dataType="org.eclipse.emf.ecore.xml.type.String" required="true"
 * @generated
 */
String getPersonName();

/**
 * Sets the value of the '{@link gcml.Person#getPersonName <em>Person Name</em>}' attribute.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @param value the new value of the '<em>Person Name</em>' attribute.
 * @see #getPersonName()
 * @generated
 */
void setPersonName(String value);

/**
 * Returns the value of the '<em><b>Person Role</b></em>' attribute.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of the '<em>Person Role</em>' attribute isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @return the value of the '<em>Person Role</em>' attribute.
 * @see #setPersonRole(String)
 * @see gcml.GcmlPackage#getPerson_PersonRole()
 * @model dataType="org.eclipse.emf.ecore.xml.type.String" required="true"
 * @generated
 */
String getPersonRole();

/**
 * Sets the value of the '{@link gcml.Person#getPersonRole <em>Person Role</em>}' attribute.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @param value the new value of the '<em>Person Role</em>' attribute.
 * @see #getPersonRole()
 * @generated
 */
void setPersonRole(String value);

/**
 * Returns the value of the '<em><b>To Is Attached</b></em>' reference.
 * <!-- begin-user-doc -->
 * <p>
 * If the meaning of the '<em>To Is Attached</em>' reference isn't clear,
 * there really should be more of a description here...
 * </p>
 * <!-- end-user-doc -->
 * @return the value of the '<em>To Is Attached</em>' reference.
 * @see #setToIsAttached(IsAttached)
 * @see gcml.GcmlPackage#getPerson_ToIsAttached()
 * @model required="true"
 * @generated
 */
IsAttached getToIsAttached();

/**
 * Sets the value of the '{@link gcml.Person#getToIsAttached <em>To Is Attached</em>}' reference.
 * <!-- begin-user-doc -->
```

```
	 * <!-- end-user-doc -->
	 * @param value the new value of the '<em>To Is Attached</em>' reference.
	 * @see #getToIsAttached()
	 * @generated
	 */
	void setToIsAttached(IsAttached value);

} // Person
```

Xcml object Model

```
//
// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference Implementation, vhudson-jaxb-ri-2.1-646
// See <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of the source schema.
// Generated on: 2008.10.21 at 10:28:09 AM EDT
//


package xcml;

import javax.xml.bind.annotation.XmlEnum;
import javax.xml.bind.annotation.XmlEnumValue;
import javax.xml.bind.annotation.XmlType;


/**
 * <p>Java class for actionType.
 *
 * <p>The following schema fragment specifies the expected content contained within this class.
 * <p>
 * <pre>
 * &lt;simpleType name="actionType">
 *   &lt;restriction base="{http://www.w3.org/2001/XMLSchema}string">
 *     &lt;enumeration value="send"/>
 *     &lt;enumeration value="doNotSend"/>
 *     &lt;enumeration value="start"/>
 *   &lt;/restriction>
 * &lt;/simpleType>
 * </pre>
 *
 */
@XmlType(name = "actionType")
@XmlEnum
public enum ActionType {

    @XmlEnumValue("send")
    SEND("send"),
    @XmlEnumValue("doNotSend")
    DO_NOT_SEND("doNotSend"),
    @XmlEnumValue("start")
    START("start");
    private final String value;

    ActionType(String v) {
    }

    public String value() {
    }

    public static ActionType fromValue(String v) {
    }

}

//
// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference Implementation, vhudson-jaxb-ri-2.1-646
// See <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of the source schema.
// Generated on: 2008.10.21 at 10:28:09 AM EDT
//


package xcml;

import javax.xml.bind.annotation.XmlEnum;
import javax.xml.bind.annotation.XmlEnumValue;
import javax.xml.bind.annotation.XmlType;
```

```
/**
 * <p>Java class for capabilityType.
 * 
 * <p>The following schema fragment specifies the expected content contained within this class.
 * <p>
 * <pre>
 * &lt;simpleType name="capabilityType">
 *   &lt;restriction base="{http://www.w3.org/2001/XMLSchema}string">
 *     &lt;enumeration value="TextFile"/>
 *     &lt;enumeration value="BinaryFile"/>
 *     &lt;enumeration value="StreamFile"/>
 *     &lt;enumeration value="NonStreamFile"/>
 *     &lt;enumeration value="AudioFile"/>
 *     &lt;enumeration value="VideoFile"/>
 *     &lt;enumeration value="AVFile"/>
 *     &lt;enumeration value="Text"/>
 *     &lt;enumeration value="LiveStream"/>
 *     &lt;enumeration value="LiveAudio"/>
 *     &lt;enumeration value="LiveVideo"/>
 *     &lt;enumeration value="LiveAV"/>
 *   &lt;/restriction>
 * &lt;/simpleType>
 * </pre>
 * 
 */
@XmlType(name = "capabilityType")
@XmlEnum
public enum CapabilityType {

    @XmlEnumValue("TextFile")
    TEXT_FILE("TextFile"),
    @XmlEnumValue("BinaryFile")
    BINARY_FILE("BinaryFile"),
    @XmlEnumValue("StreamFile")
    STREAM_FILE("StreamFile"),
    @XmlEnumValue("NonStreamFile")
    NON_STREAM_FILE("NonStreamFile"),
    @XmlEnumValue("AudioFile")
    AUDIO_FILE("AudioFile"),
    @XmlEnumValue("VideoFile")
    VIDEO_FILE("VideoFile"),
    @XmlEnumValue("AVFile")
    AV_FILE("AVFile"),
    @XmlEnumValue("Text")
    TEXT("Text"),
    @XmlEnumValue("LiveStream")
    LIVE_STREAM("LiveStream"),
    @XmlEnumValue("LiveAudio")
    LIVE_AUDIO("LiveAudio"),
    @XmlEnumValue("LiveVideo")
    LIVE_VIDEO("LiveVideo"),
    @XmlEnumValue("LiveAV")
    LIVE_AV("LiveAV");
    private final String value;

    CapabilityType(String v) {
    }

    public String value() {
    }

    public static CapabilityType fromValue(String v) {
    }

}

//
// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference Implementation,
vhudson-jaxb-ri-2.1-646
// See <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of the source schema.
// Generated on: 2008.10.21 at 10:28:09 AM EDT
//


package xcml;

import java.util.ArrayList;
import java.util.List;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlType;


/**
 * <p>Java class for connectionType complex type.
 *
 * <p>The following schema fragment specifies the expected content contained within this class.
 *
 * <pre>
 * &lt;complexType name="connectionType">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;sequence>
 *         &lt;element name="device" type="{}deviceType" maxOccurs="unbounded"/>
 *                 &lt;element  name="mediumTypeNameRef"  type="{http://www.w3.org/2001/XMLSchema}string"
maxOccurs="unbounded" minOccurs="0"/>
 *                   &lt;element   name="formTypeNameRef"   type="{http://www.w3.org/2001/XMLSchema}string"
maxOccurs="unbounded" minOccurs="0"/>
 *       &lt;/sequence>
 *       &lt;attribute name="connectionID" use="required" type="{http://www.w3.org/2001/XMLSchema}string" />
 *       &lt;attribute name="bandwidth" type="{http://www.w3.org/2001/XMLSchema}string" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "connectionType", propOrder = {
    "device",
    "mediumTypeNameRef",
    "formTypeNameRef"
})
public class ConnectionType {
```

```
@XmlElement(required = true)
protected List<DeviceType> device;
protected List<String> mediumTypeNameRef;
protected List<String> formTypeNameRef;
@XmlAttribute(required = true)
protected String connectionID;
@XmlAttribute
protected String bandwidth;

/**
 * Gets the value of the device property.
 *
 * <p>
 * This accessor method returns a reference to the live list,
 * not a snapshot. Therefore any modification you make to the
 * returned list will be present inside the JAXB object.
 * This is why there is not a <CODE>set</CODE> method for the device property.
 *
 * <p>
 * For example, to add a new item, do as follows:
 * <pre>
 *    getDevice().add(newItem);
 * </pre>
 *
 *
 * <p>
 * Objects of the following type(s) are allowed in the list
 * {@link DeviceType }
 *
 *
 */
public List<DeviceType> getDevice() {
}

/**
 * Gets the value of the mediumTypeNameRef property.
 *
 * <p>
 * This accessor method returns a reference to the live list,
 * not a snapshot. Therefore any modification you make to the
 * returned list will be present inside the JAXB object.
 * This is why there is not a <CODE>set</CODE> method for the mediumTypeNameRef property.
 *
 * <p>
 * For example, to add a new item, do as follows:
 * <pre>
 *    getMediumTypeNameRef().add(newItem);
 * </pre>
 *
 *
 * <p>
 * Objects of the following type(s) are allowed in the list
 * {@link String }
 *
 *
 */
public List<String> getMediumTypeNameRef() {
}
```

```
/**
 * Gets the value of the formTypeNameRef property.
 *
 * <p>
 * This accessor method returns a reference to the live list,
 * not a snapshot. Therefore any modification you make to the
 * returned list will be present inside the JAXB object.
 * This is why there is not a <CODE>set</CODE> method for the formTypeNameRef property.
 *
 * <p>
 * For example, to add a new item, do as follows:
 * <pre>
 *    getFormTypeNameRef().add(newItem);
 * </pre>
 *
 *
 * <p>
 * Objects of the following type(s) are allowed in the list
 * {@link String }
 *
 *
 */
public List<String> getFormTypeNameRef() {
}

/**
 * Gets the value of the connectionID property.
 *
 * @return
 *     possible object is
 *     {@link String }
 *
 */
public String getConnectionID() {
}

/**
 * Sets the value of the connectionID property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setConnectionID(String value) {
}

/**
 * Gets the value of the bandwidth property.
 *
 * @return
 *     possible object is
 *     {@link String }
 *
 */
public String getBandwidth() {
}
```

```
    /**
     * Sets the value of the bandwidth property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setBandwidth(String value) {
    }

}
```

...All other classes are similar and generated by JABX…

The Xcml class is the root of the Model and entry point

```
package xcml;

import java.util.List;


public class Xcml {
            private final UserSchema userSchema;
            private final Data data;
            private final List<ValidationError> errors;

            public Xcml(Object obj, List<ValidationError> errors) {
            }

            /**
             * @return the userSchema
             */
            public UserSchema getUserSchema() {
            }

            /**
             * @return the data
             */
            public Data getData() {
            }

            /**
             * @return the errors
             */
            public List<ValidationError> getErrors() {
            }

            /**
             * @return
             */
            public boolean containsErrors() {
            }
}

package xcml;

import java.io.File;
import java.io.StringReader;
import java.io.StringWriter;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;
import javax.xml.validation.SchemaFactory;

import org.w3c.dom.Node;
import org.xml.sax.SAXException;


public class XcmlFactory extends ObjectFactory {
		private final JAXBContext jaxbContext;
		private final Unmarshaller unmarshaller;
		private final Marshaller marshaller;
		private final XcmlValidationEventHandler evtHandler;

		/**
		 * @return
		 */
		private static File getSchemaFile() {
		}

		/**
		 * @throws JAXBException
		 * @throws SAXException
		 */
		public XcmlFactory() throws JAXBException, SAXException {
		}

		/**
		 * @param xml
		 * @return
		 * @throws JAXBException
		 */
		public synchronized Xcml readXcml(File xml) throws JAXBException {
		}

		/**
		 * @param xml
		 * @return
		 * @throws JAXBException
		 */
		public synchronized Xcml readXcml(Node xml) throws JAXBException {
		}

		/**
		 * @param rootElement
		 * @return
		 * @throws JAXBException
		 */
		public synchronized String writeXcml(Xcml rootElement) throws JAXBException {
		}
}

package xcml;

import java.util.LinkedList;
```

```java
import java.util.List;

import javax.xml.bind.ValidationEvent;
import javax.xml.bind.ValidationEventHandler;
import javax.xml.bind.ValidationEventLocator;

public class XcmlValidationEventHandler implements ValidationEventHandler {
		private List<ValidationError> errors;

		/**
		 *
		 */
		public XcmlValidationEventHandler() {
		}

		/* (non-Javadoc)
		 * @see javax.xml.bind.ValidationEventHandler#handleEvent(javax.xml.bind.ValidationEvent)
		 */
		public boolean handleEvent(ValidationEvent ve) {
		}

		public List<ValidationError> getErrors() {
		}
}

package xcml;


/** Similar to the XCML classes implemented by Team 1
 * @
 *
 */
public interface XCMLVisitor {


		/**
		 * @param node
		 * @return
		 */
		public Object visit(ConnectionType node);

		/**
		 * @param node
		 * @return
		 */
		public Object visit(Data node);

		/**
		 * @param node
		 * @return
		 */
		public Object visit(DeviceType node);

		/**
		 * @param node
		 * @return
		 */
		public Object visit(FormType node);
	/**
	 * @param node
	 * @return
	 */
	public Object visit(FormTypeType node);

	/**
	 * @param node
	 * @return
	 */
	public Object visit(IsAttachedType node);

	/**
	 * @param node
	 * @return
	 */
	public Object visit(MediumType node);

	/**
	 * @param node
	 * @return
	 */
	public Object visit(MediumTypeType node);

	/**
	 * @param node
	 * @return
	 */
	public Object visit(PersonType node);

	/**
	 * @param node
	 * @return
	 */
	public Object visit(Xcml node);

	/**
	 * @param node
	 * @return
	 */
	public Object visit(UserSchema node);
}
```

Repository

```
/**
 * @author Leandro Wong
 * the Repository contains the saved models
* *
 */

package cme.repository;

import java.util.ArrayList;

public class Repository {
	private ArrayList<ModelMetadata> models;
	private ArrayList<User> users;
	private static Repository instance;

	private Repository(){
		//requires use of instance method
	}

	/**
	 * @return the singleton instance
	 */
	public static Repository getInstance() {
	}
	private static void initializeInstance(){
		// do some initialization based on configuration
		// for example the general path to the repository
		// the repository type (if is a file server or
		// a database)
	}

	public ArrayList<ModelMetadata> getModels() {
	}
	public void setModels(ArrayList<ModelMetadata> models) {
	}
	public ArrayList<User> getUsers() {
	}
	public void setUsers(ArrayList<User> users) {
	}
}


package cme.repository;

public class User {
	private String userName;
	private String password;
	private String fullName;
	private UserType userType = UserType.Modeler;

	public UserType getUserType() {
	}
	public void setUserType(UserType userType) {
	}
	public String getUserName() {
	}
	public void setUserName(String userName) {
	}
	public String getPassword() {
	}
	public void setPassword(String password) {
        }
        public String getFullName() {
        }
        public void setFullName(String fullName) {
        }

        public enum UserType{
                Modeler,
                Administrator
        }
}

package cme.repository;

public class ModelMetadata {
        public String getName() {
        }

        public void setName(String name) {
        }

        public String getDescription() {
        }

        public void setDescription(String description) {
        }
        public String getGcmlPath() {
        }

        public void setGcmlPath(String gcmlPath) {
        }

        public String getGcmlLayoutPath() {
        }

        public void setGcmlLayoutPath(String gcmlLayoutPath) {
        }

        public String getXcmlPath() {
        }

        public void setXcmlPath(String xcmlPath) {
        }

        public String getId() {
        }

        public void setId(String id) {
        }

        public Format getFormat() {
        }

        public void setFormat(Format format) {
        }

        public enum Format {
                Xcml, Gcml, Both
        }
```

```
}
```

UCI (Transformation and Validation)

```
/**
 * @author Lazaro Pi
 * the Filter interface
 * provides the interfaces that all the filters will inherit from
 *
 */

package cme.uci;

import java.util.ArrayList;


public abstract class Pipeline {


	public String Execute() {

	}

	private ArrayList<Filter> filters = new ArrayList<Filter>();

	protected ArrayList<Filter> getFilters() {
	}
}

package cme.uci;

/**
 * @author Lazaro Pi
 * the Filter interface
 */
public interface Filter {

	/**
	 * Gets the path in.
	 *
	 * @return the path in.
	 */
	String getPathIn();

	/**
	 * Sets the path in.
	 *
	 * @param value
	 *            the path in
	 */
	void setPathIn(String value);

	/**
	 * Gets the path out.
	 *
	 * @return the path out.
	 */
	String getPathOut();

	/**
```

```
         * Sets the path out.
         *
         * @param value
         *          the path out
         */
        void setPathOut(String value);

        /**
         * Executes this instance.
         */
        void Execute();
}

package cme.uci;


public abstract class AbstractFilter implements Filter{

        /*
         * (non-Javadoc)
         *
         * @see cme.transform.Filter#getPathIn()
         */
        public String getPathIn() {
        }

        /*
         * (non-Javadoc)
         *
         * @see cme.transform.Filter#setPathIn(java.lang.String)
         */
        public void setPathIn(String in) {
        }

        /*
         * (non-Javadoc)
         *
         * @see cme.transform.Filter#getPathOut()
         */
        public String getPathOut() {
        }

        /*(non-Javadoc)
         * @see cme.transform.Filter#setPathOut(java.lang.String)
         */
        public void setPathOut(String out) {
        }

        /**
         * the path in for this filter.
         */
        private String _pathIn;

        /**
         * the path out for this filter.
         */
        private String _pathOut;
}
```

```
package cme.uci.transform;

import bin.gcml.Gcml;
import bin.xcml.Xcml;

public class Transform {
	public static void fromXcmlToGcml(String xcmlPath, String gcmlPath,
					String gcmlLayoutPath) {
		//do transformation here
	}
	public static void fromGcmlToXcml(String gcmlPath, String gcmlLayoutPath,
					String xcmlPath) {
		//do transformation here
	}
	public static Xcml toXcml(Gcml gcmlInstance){
		//return in memory
	}
	public static Gcml toGcml(Xcml xcmlInstance){
		//return in memory
	}
}

package cme.uci.transform;

import java.io.File;
import java.util.ArrayList;

import cme.uci.Filter;
import cme.uci.Pipeline;
import cme.uci.filters.SerializerFilter;
import cme.uci.filters.TransformFilter;

/**
 * Represents the Transformations as a linear pipeline where all the steps are
 * executed in order and the input of one step is the output of the previous
 * step. The last output is returned to the client that called the execute
 * method.
 *
 * @author Lazaro Pi
 */
public class TransformPipeline extends Pipeline {
	/**
	 * Initializes a new instance of the Pipeline class. Private in order to
	 * force the use of the factory method.
	 */
	private TransformPipeline() {
		// requires factory method
	}


	public static TransformPipeline createGcmlToXcmlPipeline(String pathIn) {
		// creates the pipeline

		// creates the working directory

		// adds all the filters to the pipeline

		// returns the pipeline
	}

	public static TransformPipeline createXcmlToGcmlPipeline(String pathIn) {
		// creates the pipeline

		// creates the working directory

		// adds all the filters to the pipeline

		// returns the pipeline
	}

}

package cme.uci.validate;

import bin.gcml.Gcml;
import bin.xcml.Xcml;

public class Validate {
	public static boolean validateGcml(String gcmlPath, String gcmlLayoutPath) {
		// validate
	}

	public static boolean validateXcml(String xcmlPath) {
		// validate
	}

	public static boolean validateXcml(Xcml xcmlInstance) {
		// validate
	}

	public static boolean validateGcml(Gcml gcmlInstance) {
		// validate
	}
}

package cme.uci.validate;

import cme.uci.Pipeline;

public class ValidationPipeline extends Pipeline {
	/**
	 * Initializes a new instance of the Pipeline class. Private in order to
	 * force the use of the factory method.
	 */
	private ValidationPipeline() {
		// requires factory method
	}

	public static ValidationPipeline createPipeline(String pathIn) {
		// creates the pipeline

		// TODO: adds all the filters to the pipeline

		// returns the pipeline

	}
```

```
}


package cme.uci.filters;

import java.util.HashMap;
import java.util.Map;

import cme.uci.AbstractFilter;


/**
 * @author Lazaro Pi Represents one step in the pipeline of transformations.
 *         Transforms the input into an output file by using the specified XSLT
 *         and code extensions.
 */
public class TransformFilter extends AbstractFilter {

        /**
         * the transform parameters for this filter
         */
        private TransformParms parms;

        /**
         * @param parms
         *            the parameters
         */
        public void setParms(TransformParms parms) {
        }

        /**
         * @return the parameters
         */
        public TransformParms getParms() {
        }

        /*
         * (non-Javadoc)
         *
         * @see cme.transform.Filter#Execute()
         */
        public void Execute() {
        }

        /**
         * Transforms the input into an output file by using the specified
         * parameters.
         *
         * @param pathIn
         *            the path in
         * @param pathOut
         *            the path out
         * @param parms
         *            the parameters
         */

        public static TransformFilter createGcmlToXcmlFilter(String pathIn,
                        String pathOut) {
        }
	public static TransformFilter createXcmlToGcmlFilter(String pathIn,
					String pathOut) {
	}

}
```

```
package cme.uci.filters;

import cme.uci.AbstractFilter;

public class CheckSemanticRulesFilter extends AbstractFilter {

	@Override
	public void Execute() {
		// TODO Auto-generated method stub
	}

	/**
	 * Creates the semantic rules filter
	 * @param pathIn the path in.
	 * @param pathOut the path out.
	 * @return the created filter
	 */
	public static CheckSemanticRulesFilter createCheckModelSchemaFilter(String pathIn,
					String pathOut) {
	}
}
```

…More Filter classes to accomplish transformation and validation…
…they all have an execute method…

Administration and Security

```
/* This package is in charge of the access Rights
 *
 *Team 2
*/

package cme.admin;

import java.util.List;

import cme.repository.Repository;
import cme.repository.User;

public class SecurityContext {
		private static User currentUser;

		public static boolean Login(String userName, String password){
				//checks credentials and sets the current user
				//returns true if valid login
		}

		public static void Logout(){
		}
}


package cme.admin;

public class Crypto {

		/**
		 * Encrypts the decrypted string
		 * @param original
		 * @return the encrypted String
		 */
		public static String encrypt(String original){
		}
		/**
		 * Decrypts the encrypted String
```

```
            * @param encrypted
            * @return the decrypted string
            */
           public static String decrypt(String encrypted){
           }
}
```

10.6. Appendix F – Documented Code for Test Drivers

/*This class contains the test driver for Transform Java. The driver works by defining the args that the class is run with   author Team 2. */

```
package cme.uci.transform;

import java.io.File;
import java.io.FileNotFoundException;

import javax.xml.transform.TransformerException;

import gcml.Gcml;
import xcml.*;


public class Transform {
   public static void fromXcmlToGcml(String xcmlPath, String gcmlPath,
         String gcmlLayoutPath) {
      // do transformation here
      XcmlProvider provider = new XcmlProvider();
      Xcml xcml = provider.getObjects(new File(xcmlPath));

   }

   public static String fromGcmlToXcml(String gcmlPath, String gcmlLayoutPath,
         String xcmlPath) throws FileNotFoundException, TransformerException {
      XslTransform.transform(gcmlPath, Transform.class.getResource(
              "FromGcmlToXcml.xslt").getPath(), xcmlPath);

      // check that it can be read as XCML
      XcmlProvider provider = new XcmlProvider();
      Xcml xcml = provider.getObjects(new File(xcmlPath));

      if (xcml != null)
         return xcml.getUserSchema().getCommunicationID();
      else
         return "";
   }

   private static Xcml toXcml(Gcml gcmlInstance) {
      // return in memory
      return null;
   }

   private static Gcml toGcml(Xcml xcmlInstance) {
      // return in memory
      return null;
   }

   public static void main(String[] args) throws FileNotFoundException,
         TransformerException {
      // sample arguments
      // -useCurrentFolder -option(fromGcmlToXcml) gcmlSample.gcml
      //         gcmlSample.gcml_diagram out.xcml

      if (args.length == 4) {
         if ("-option(fromGcmlToXcml)".equals(args[0])) {
              Transform.fromGcmlToXcml(args[1], args[2], args[3]);
         } else if ("-option(fromXcmlToGcml)".equals(args[0])) {
              Transform.fromXcmlToGcml(args[1], args[2], args[3]);
         }
      } else if (args.length == 5 && "-useCurrentFolder".equals(args[0])) {
         if ("-option(fromGcmlToXcml)".equals(args[1])) {
              Transform.fromGcmlToXcml(Transform.class.getResource(args[2])
                      .getPath(), Transform.class.getResource(args[3])
                      .getPath(), Transform.class.getResource(args[4])
                      .getPath());
            } else if ("-option(fromXcmlToGcml)".equals(args[1])) {
                Transform.fromXcmlToGcml(Transform.class.getResource(args[2])
                        .getPath(), Transform.class.getResource(args[3])
                        .getPath(), Transform.class.getResource(args[4])
                        .getPath());
            }
        } else {
            System.err
                    .println("Usage: java Transform <-useCurrentFolder> [-
option(fromGcmlToXcml)] [gcmlPath] [gcmlLayoutPath] [xcmlPath]");
            System.exit(1);
        }

    }
}
```

10.7. Appendix G – Diary of Meetings and Tasks

Meeting 1

Date:	8/27/2008
Moderator:	Lazaro
Place:	Classroom
From:	9:10 PM
To:	10:00 PM
Duration:	1 hour
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	Presentation of members
	Exchange contact information
	Overview of the project
Discussion Topics:	Overview of the entire project
Assignments:	Review websites, old projects

Meeting 2

Date:	8/30/2008
Moderator:	Leandro
Place:	Graduate Lab
From:	1:00 PM
To:	3:00 PM
Duration:	2 hours
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	Overview of the project
	Use cases
Discussion Topics:	Project, Use Cases
Assignments:	Use Cases

Meeting 3

Date:	9/03/2008
Moderator:	Andrew
Place:	Andrew Office
From:	9:00 PM
To:	10:00 PM
Duration:	1 hour
Participants:	Andrew, Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto, Andrew, Frank
Members Late:	None
Agenda:	Andrew Presentation
	Overview of the project
	Project Requirements
Discussion Topics:	Functionality and Limitations
Assignments:	Continue working on previous assignments

Meeting 4

Date:	9/06/2008
Moderator:	Manasa
Place:	Graduate Lab
From:	9:00 AM
To:	11:00 AM
Duration:	2 hours
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	Project Requirements Review of Use Cases Review of scenarios Overview of the project GCML XCML How to store models How to load and display models Transformation
Discussion Topics:	Transform models in GCML to schemas in XCML, storing, saving and loading models
Assignments:	Project requirements, Use Cases, scenarios Study GCML, study XCML, make sure Eclipse is working in all group member's computers, continue working on assignments from previous week

Meeting 5

Date:	9/10/2008
Moderator:	Marc
Place:	Graduate Lab
From:	3:00 PM
To:	5:00 PM
Duration:	2 hours
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	SRD Cost of the Project Identification of Actors Uses Cases Revision Redefining Uses Cases Use Cases Interaction Project Organization
Discussion Topics:	Use Cases, SRD, functional requirements, non-functional requirements, Project

	Scheduling, glossary, use interfaces, project costing estimation, installing software to calculate cost, Current System, Scope of the System, Definitions
Assignments:	Re assignment of Use Cases, Continue working on assignments from previous week, Review of SRD requirements

Meeting 6

Date:	9/13/2008
Moderator:	Jorge
Place:	Under Graduate Lab
From:	1:00 PM
To:	4:00 PM
Duration:	3 hours
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	SRD in general Assignment for SRD Installation of COCOMO II Costing Estimation
Discussion Topics:	SRD, Eclipse and assignments, COCOMO, cost of the Project
Assignments:	Revision of Use Cases, Continue working on assignments from previous week, Review of SRD requirements

SRD:

Introduction: Roberto
Chapter 1: Roberto
Chapter 2: Sandeep
Chapter 3: Leandro
Chapter 4: Lazaro, Manasa, Jorge, Marc
Chapter 5: Roberto
Chapter 6: Sandeep
Chapter 7: 7.1 Leandro -- 7.2 Roberto -- 7.2 Marc

Meeting 7

Date:	9/17/2008
Moderator:	Lazaro
Place:	Graduate Lab
From:	2:00 PM
To:	5:00 PM
Duration:	3 hours
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	SRD Revision of assignments from previous meeting Questions and Clarification Use Cases interaction
Discussion Topics:	Assigned sections from the SRD
Assignments:	Review each section of the SRD and continue working on the SRD assignments

Meeting 8

Date:	9/24/2008
Moderator:	Leandro
Place:	Graduate Lab
From:	3:00 PM
To:	5:00 PM
Duration:	2 hours
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	SRD – user interfaces SRD – Dairy of meetings SRD - Project scheduling Presentation 1 requirements Review of COCOMO cost results
Discussion Topics:	Content and format of the Project Schedule, content and format of the Dairy of Meetings. Presentations of user interfaces Checklists Revision, status and progress and re-assignments of activities, scenarios, sequence diagrams
Assignments:	Work on assignments for SRD

Meeting 9

Date:	10/1/2008
Moderator:	Sandeep
Place:	Graduate Lab
From:	1:00 PM
To:	4:00 PM
Duration:	3 hours
Participants:	Lazaro , Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	SRD – Hardware and Software requirements
	Presentation 1
	Sequence Diagrams
	Review for the Exam
Discussion Topics:	Static Model, Dynamic Model, SRD Document Revision, status and progress. Review of the Project for the exam,
	Use Cases Revision, status and progress
Assignments:	None

Meeting 10

Date:	10/4/2008
Moderator:	Leandro
Place:	Graduate Lab
From:	08:00 AM
To:	11:00 AM
Duration:	3 hours
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	SRD - Diagrams
	Review of Presentation 1
	Practice Presentation 1
	SRD – Review all the sections that are finished.
Discussion Topics:	Content of Presentation 1 and sequence diagrams.
	We are not satisfy with 2 of the Use Cases.
	We have to re-do 2 of the Use Cases.
Assignments:	Finish assignments for SRD

Meeting 11

Date:	10/5/2008
Moderator:	Lazaro
Place:	Graduate Lab
From:	2:00 PM
To:	3:00 PM
Duration:	1 hour
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	SRD – Review all the Appendixes. Project Schedule Dairy of Meetings User Interface Designs
Discussion Topics:	Schedule and interface designs
Assignments:	Complete and finish the interfaces designs, finish the Gantt chart for the Project

Meeting 12

Date:	10/6/2008
Moderator:	Roberto
Place:	Graduate Lab
From:	4:00 PM
To:	7:00 PM
Duration:	3 hours
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	Final Review of the SRD – Practice Presentation 1
Discussion Topics:	Changes to done to the SRD
Assignments:	Complete, revise and finish all the sections of the SRD

Meeting 13

Date:	10/7/2008
Moderator:	Sandeep
Place:	Graduate Lab
From:	12:00 PM
To:	4:00 PM
Duration:	4 hours
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	Presentation of the SRD by section – before printing final copy
Discussion Topics:	SRD is approved by the all members of the group
Assignments:	Printing final copy according to the specifications

Meeting 14

Date:	10/8/2008
Moderator:	Lazaro
Place:	Classroom
From:	9:10 PM
To:	10:00 PM
Duration:	1 hour
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	**Previous Deliverable Flaws Discussed** Roles for the Design Phase Defined Identified the UseCases that need to be implemented
Discussion Topics:	Choosing the most relevant Use Cases to be implemented
Assignments:	Sequence Diagram

Meeting 15

Date:	10/18/2008
Moderator:	Manasa
Place:	Graduate Lab
From:	1:00 PM
To:	3:00 PM
Duration:	2 hours
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	Sequence Diagrams Object Diagrams Minimal Class Diagram
Discussion Topics:	Project,UML diagrams
Assignments:	UML diagrams

Meeting 16

Date:	10/22/2008
Moderator:	Andrew
Place:	Andrew Office
From:	9:00 PM
To:	10:00 PM
Duration:	1 hour
Participants:	Andrew, Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto, Andrew, Clarke
Members Late:	None
Agenda:	Andrew Discussion of the USECASES to be Implemented Decided on the format of the diagrams Project Requirements
Discussion Topics:	Discussion of the Class diagrams and UML stuff
Assignments:	Continue working on previous assignments

Meeting 17

Date:	10/25/2008

Moderator:	Leandro
Place:	Graduate Lab
From:	9:00 AM
To:	11:00 AM
Duration:	2 hours
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	Project Requirements Review of Package Diagrams Review of Minimal and Detailed Class Diagrams Identified the Design Pattern and the Architectutal Pattern GCML XCML How to store models How to load and display models Transformation
Discussion Topics:	Transform models in GCML to schemas in XCML, storing, saving and loading models
Assignments:	Project requirements, Use Cases, scenarios Work on the implementation of the design and architectural pattern

Meeting 18

Date:	9/10/2008
Moderator:	Marc
Place:	Graduate Lab
From:	3:00 PM
To:	5:00 PM
Duration:	2 hours
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	SRD Cost of the Project Identification of Actors Uses Cases Revision Redefining Uses Cases Use Cases Interaction Project Organization

Discussion Topics: Use Cases, SRD, functional requirements, non-functional requirements, Project Scheduling, glossary, use interfaces, project costing estimation, installing software to calculate cost, Current System, Scope of the System, Definitions

Assignments: Re assignment of Use Cases, Continue working on assignments from previous week, Review of SRD requirements

Meeting 19

Date:	11/1/2008

Moderator:	Jorge
Place:	Under Graduate Lab
From:	1:00 PM
To:	4:00 PM
Duration:	3 hours
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep, Roberto
Members Late:	None
Agenda:	Design Document Review Discussed and did the OCL Focused on the implementation Details of the Project Stat Chart diagrams, identification of Control objects
Discussion Topics:	Control Objects, Entity Classes (Persistent Data Model), OCL
Assignments:	OCL, State Chart Diagrams and Persistent Data Management

DD:

Introduction: Roberto
Chapter 1: Sandeep, Roberto
Chapter 2: Lazaro, Manasa, Leandro
Chapter 3: Jorge, Sandeep,Marc, Leandro
Chapter 4: Roberto
Chapter 5: Roberto,Manasa ,Lazaro, Marc,Leandro

Meeting 20

Date:	11/12/2008
Moderator:	Lazaro
Place:	Classroom
From:	9:00 PM
To:	10:00 PM
Duration:	1 hour
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep
Members Late:	None
Agenda:	Discuss implementation
Discussion Topics:	Existent software, work left to do
Assignments:	Developers need to get the existing to run in Eclipse with the correct libraries

Meeting 21

Date:	11/21/2008
Moderator:	Lazaro
Place:	Library 2nd Floor
From:	5:00 PM
To:	7:00 PM
Duration:	2 hour
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep
Members Late:	None
Agenda:	Previous Deliverable Flaws Discussed Roles for the Final Deliverable Defined
Discussion Topics:	Second deliverable and Final deliverable
Assignments:	Prepare next meeting : test cases

Meeting 22

Date:	11/23/2008
Moderator:	Manasa
Place:	Library
From:	2:00 PM
To:	3:00 PM
Duration:	1 hour
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep
Members Late:	None
Agenda:	Outline of the test cases
Discussion Topics:	Choosing the most relevant test cases for the system
Assignments:	Write the complete test cases

Meeting 23

Date:	11/25/2008
Moderator:	Jorge
Place:	Graduate Lab
From:	6:00 PM
To:	7:30 PM
Duration:	1.5 hour
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep
Members Late:	None
Agenda:	Outline of the test cases
Discussion Topics:	Choosing the most relevant test cases for the system
Assignments:	Write the complete test cases

Meeting 24

Date:	11/26/2008
Moderator:	Lazaro
Place:	Classroom
From:	2:00 PM
To:	3:00 PM
Duration:	1 hour
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep
Members Late:	None
Agenda:	Update on the work on test cases Update on the implementation
Discussion Topics:	Test cases, code
Assignments:	Continue development

Meeting 25

Date:	11/30/2008
Moderator:	Sandeep
Place:	Grad Lab

From:	2:00 PM
To:	5:00 PM
Duration:	3 hour
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep
Members Late:	None
Agenda:	Finalizing the document Preparing presentation
Discussion Topics:	Document, implementation
Assignments:	Learn the presentation

Meeting 26

Date:	12/02/2008
Moderator:	Sandeep
Place:	Grad Lab
From:	6:00 PM
To:	7:00 PM
Duration:	1 hour
Participants:	Lazaro, Leandro, Manasa, Marc, Jorge, Sandeep
Members Late:	None
Agenda:	Rehearsing the presentation
Discussion Topics:	Presentation
Assignments:	Final preparation. Print Final Document.