SOFTWARE FOR VISUALIZATION AND COORDINATION OF THE DISTRIBUTED SIMULATION MODELING PROCESS

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Yudi Xue

©Yudi Xue, April/2012. All rights reserved.

PERMISSION TO USE

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

ABSTRACT

Simulation modeling projects commonly involve distributed team collaboration. It is currently difficult to perform collaboration in distributed modeling process for two reasons: 1) Simulation modeling in general requires modelers to manage complexities (such as tracking model revisions, recording scenario assumptions and organizing external artifacts) related to the model. 2) Distributed collaboration requires collaborators to maintain change awareness. While proper information technology support is known to lessen the difficulties of collaborations, there is limited software support for complexity management in generic modeling process and change awareness in distributed collaboration, therefore require tremendous amount of effort in management and communication. This thesis describes a new system that supports distributed modeling process. The system provides modeling repositories to help manage modeling complexities and a visual workspace to provide change awareness information. The system has been shown to substantially reduce modeling effort in distributed modeling, is extensible and easy to use.

ACKNOWLEDGEMENTS

I would like to thank my thesis advisors, Carl Gutwin and Nate Osgood, for their help and extreme patience during the course of my thesis program.

Additionally, I would like to give my extreme thanks to my wife, my parents and friends for all the love and support they have continually and unconditionally given to me throughout the years. Their support is fundamental to all my successes.

Dedicated to my parents – Xue, Zheng and Zheng, Shaoping.

Contents

Permission to Use **i**

Abstract **ii**

Acknowledgements **iii**

Contents **v**

List of Tables **vii**

List of Figures **viii**

List of Abbreviations **ix**

1 Introduction **1**

1.1 Problem . . . 3
1.2 Goals and Motivation . . . 4
1.3 Solution . . . 4
1.4 Steps in the Solution . . . 5
1.4.1 Identification of System Requirements . . . 6
1.4.2 Development of a Modeling Process Hierarchy . . . 6
1.4.3 Development of the Visual Workspace . . . 7
1.4.4 Development of Model Package Support . . . 7
1.5 Evaluation . . . 8
1.6 Contributions . . . 8
1.7 Thesis Outline . . . 8

2 Background and Related Work **10**

2.1 The Modeling Process . . . 10
2.1.1 Theory of Modeling Process . . . 10
2.1.2 Overview of System Dynamics Modeling . . . 10
2.1.3 Overview of Agent-based Modeling . . . 13
2.1.4 Modeling Process Hierarchy . . . 15
2.1.5 Modeling Software . . . 16
2.2 Awareness . . . 16
2.2.1 What Information Makes Up Change Awareness? . . . 18
2.2.2 How is Change Awareness Generally Supported in Collaboration? . . . 22
2.3 Component Software . . . 23
2.3.1 Equinox OSGi . . . 23

3 System Requirements **24**

3.1 System Integration for the Modeling Process Hierarchy . . . 24
3.2 Modeling Process Support . . . 25
3.2.1 Modeling Version Control . . . 26
3.2.2 Scenario Bookkeeping . . . 26
3.2.3 Linking External Artifacts . . . 26
3.2.4 Simulation Support . . . 26
3.3 User Interface In SILVER . . . 27
3.3.1 Main Interface . . . 27
3.3.2 Dedicated Collaborative Workspace for Change Awareness . . . 28

3.4 Change Awareness Information in Each User Interface Component . . . 30

4 System Design and Implementation 34
4.1 New Support Features Provided by SILVER . . . 35
4.1.1 System Highlights . . . 36
4.1.2 Distribution Architecture . . . 36
4.2 System Infrastructure . . . 38
4.2.1 General Design Goals . . . 38
4.2.2 System Implementation Goals . . . 39
4.2.3 Business Logic Component Design . . . 40
4.2.4 Storage Component Design . . . 43
4.2.5 Database Design . . . 44
4.2.6 Change Logging . . . 46
4.2.7 Internationalization . . . 46
4.3 UI Component Design . . . 46
4.3.1 Visual Workspace Component Design . . . 47
4.4 Simulation Software Support . . . 49
4.4.1 Vensim Support . . . 49
4.4.2 AnyLogic Support . . . 50

5 The Visual Workspace 53
5.1 The Visual Workspace . . . 53
5.1.1 Visualizing the Model Hierarchy . . . 53
5.1.2 Displaying ChangeSet . . . 55
5.2 ChangeSet Editor . . . 56

6 The Modeling Process with SILVER 57
6.1 Working with an Anylogic model . . . 57
6.2 Working with a Vensim Model . . . 59
6.3 Working with Models Stored Remotely . . . 61
6.3.1 Adding External Documents . . . 62

7 Evaluation 63
7.1 Evaluation Process . . . 64
7.2 Extensibility . . . 65
7.2.1 Example to Extend SILVER . . . 66
7.3 Effectiveness from the User's Perspective . . . 66
7.3.1 Effectiveness of Modeling Process Hierarchy . . . 68
7.3.2 Effectiveness of Change Awareness . . . 69
7.4 Limitations . . . 69

8 Conclusion 72
8.1 Summary of Research and Contributions . . . 72
8.2 Future Work . . . 74

References 76

A Database ER Diagram 79

B SILVER Evaluation Form 80

List of Tables

2.1 Informational elements related to the 'where' aspect of change awareness [56] 19
2.2 Informational elements related to the 'who' aspect of change awareness [56] 20
2.3 Informational elements related to the 'what' aspect of change awareness [56] 20
2.4 Informational elements related to the 'how' aspect of change awareness [56] 21
2.5 Informational elements related to the 'when' aspect of change awareness [56] 21
2.6 Informational elements related to the 'why' aspect of change awareness [56] 22

3.1 Supported Actions in MPH 25
3.2 Supported change awareness in the information panel 31
3.3 Supported change awareness in the visual workspace 32
3.4 Supported change awareness in the change log panel 33

4.1 Database Table Specification 52

7.1 Change Awareness Evaluation 68

List of Figures

2.1 Iterative System Dynamics Modeling Process from [55] 11
2.2 The modeling cycle from Railsback et al 2011 14

3.1 Overview of the SILVER User Interface 27
3.2 SILVER Visual Workspace 29
3.3 SILVER Change History Panel 30

4.1 A High-Level View of the SILVER Architecture 35
4.2 Example request flow in the business logic component 40
4.3 Business logic component design 42
4.4 Storage component architecture 44
4.5 Database Entity Relationship 45
4.6 User Interface Component Design 46
4.7 The Infovis Model in Prefuse, (figure from Heer 2005) 48
4.8 Infovis Model in the Visual Workspace 48
4.9 Anylogic Support in SILVER 51

5.1 The Visual Workspace 54
5.2 Highlighting Changes 55
5.3 ChangeSet editor 56

6.1 Create an Anylogic model 58
6.2 Import an Anylogic model to SILVER 59
6.3 Publish an Vensim model 60

7.1 Create test plugin 63
7.2 Import base plugin in test plugin 64
7.3 Import test plugin in SILVER 65
7.4 SILVER evaluation analysis 67

A.1 Database Entity Relationship 79

LIST OF ABBREVIATIONS

LOF	List of Figures
LOT	List of Tables
ACID	atomicity, consistency, isolation, durability
JDBC	Java Database Connectivity
OSGi	Open Services Gateway initiative
RCP	Rich Client Platform
JTA	Java Transaction API
API	Application Programming Interface
MPH	Modeling Process Hierarchy
CSCW	Computer Supported Cooperative Work
GUI	Graphical User Interface
IDE	Integrated Development Environment
UI	User Interface
SQL	Structured Query Language
XML	Extensible Markup Language
UUID	Universally Unique Identifier
JVM	Java Virtual Machine
UTF-8	UCS Transformation Format 8-bit
ANSI	American National Standards Institute

CHAPTER 1

INTRODUCTION

The simulation modeling process (henceforth termed the “modeling process”), is the process of stating hypotheses and gaining understanding regarding how the interaction of parts of a system give rise to the behavior of the system as a whole. It involves building a model that is a simplified representation of a system and manipulating it in a simulation that runs over time based on defined rules and interactions within the model. The modeling process is valuable because it allows stake-holders to rise above the welter of operation detail in a complex system, explore various hypotheses and seek answers to “what-if” questions. Regardless of the scope of model or the field of application, *modelers* adhere to an experimental process while performing the modeling process to improve their understanding about complex systems. *Modelers* – i.e model builders – are skilled professionals who are responsible for building a model based on a client’s domain system and requirements, and for performing analysis to address the client’s concerns. *Clients* are stakeholders who demand inexpensive and rigorous simulation of a system to investigate ideas and to understand the system. As a scientific methodology, simulation modeling is notable for the depth and flexibility of evaluation and revision performed, and – in many cases – by an insistence upon empirical evidence to support hypotheses and formulated questions. As a result, the simulation modeling process is iterative in nature to reflect insights from evaluation and testing. The iteration cycle in the modeling process involves several steps: hypothesis formulation and “problem articulation”, model building, model testing and model evaluation [31].

Previous researchers have described the theory for modeling process iteration in the following fashion:

“Model revision may consist of changes in system boundary, level of aggregation, or detailed formulation. Revisions often result in greater model complexity, but realistic detail should not be introduced at the expense of comprehensibility. The iterative process may, in theory, continue as long as the model fails to satisfy some evaluative criterion [51, 52]."

A modeling process involving a diverse team of stakeholders is a form of collaboration for the purpose of creating and refining a model. Similar collaboration in artifact creation is common in many fields, such as co-authoring papers or developing medium to large scale software systems [44]. In the context of collaboration, steps in the modeling process are managed across geographically distributed organizations and teams. I refer to a modeling process performed through distributed collaboration as *collaborative modeling*. I define *remote collaborative modeling* as performing iterative modeling in teams of modelers, clients and other stakeholders, some or all of whom are spatially distributed.

Increases in computing power and communications infrastructure allow simulation modeling to involve large teams across regions or organizations. In these situations, it has been difficult to perform collaborative modeling [45]. When multiple people contribute during collaborative modeling, changes can quickly spiral out of control because changes made by one person can affect the work of others and add to the burden of tracking changes. As pointed out by Osgood, the difficulties in collaborative modeling are mainly involve model version control, scenario bookkeeping, linking artifacts to external supporting artifacts, recording modeling intentions and expressing high-level scenario structure [48]. The difficulties encountered in collaborative modeling require collaborators to put effort into collecting and recording information about version-specific structural changes, recording assumptions underlying scenario-specific simulations, and maintaining change awareness about group activities over time. *Change awareness* is an extention of Gutwin's concept of workspace awareness [28]. It represents the ability for individuals to track asynchronous changes made in collaboration [56]. Change awareness involves information about who is participating in the project, what have they been working on, and what changes they have made.

This thesis is concerned with providing tools to support the modeling process in distributed collaboration. The provision of this support has been accomplished in several steps. The first step involves building a software system which constructs an organized, virtual hierarchy to support the general steps that are carried out in the modeling process.

We term the virtual hierarchy provided in the system as *Modeling Process Hierarchy*. The modeling process hierarchy consists of mechanisms for model version control, scenario exploration and the attachment of documentation. Next, the system provides change awareness information based on the modeling process hierarchy. Finally, the system has been developed such that it can be easily extended to work with multiple types of modeling software; it currently directly interprets models created by Vensim and Anylogic, but also permits storage of models and scenario information created by other systems. Therefore, the scope of models this system can be supported are aggregate models built by Vensim and Agent-based models built by Anylogic [11, 1].

1.1 Problem

The problem addressed by this thesis is that modelers have difficulty in conducting distributed collaboration in the modeling process.

Conducting the modeling process involving distributed collaboration is difficult because:

(i) Such collaboration traditionally lacks an explicit virtual structure to organize the large amount of information generated during the modeling process.

(ii) Maintaining change awareness is necessary but difficult in collaborative modeling.

There are several factors that must be considered when supporting collaborative modeling. First, system support needs to reduce information complexity involved in the modeling process. This includes explicitly representing and maintaining the relationships between each step of the modeling process. For example, the output from a scenario simulation is specific to a model version and a particular parameter set. Second, the system needs to support two different contexts of use: in the distributed modeling process, the individual modeling process and the collaborative modeling process. This is because a modeler needs a way to keep certain work in private while also contributing to the team. The system should provide modelers with a private modeling process hierarchy for individual purposes and access the shared modeling process hierarchy for collaboration purposes. Finally, as with distributed system, the technical issue of maintaining a stable and scalable data transfer mechanism over the Internet is essential.

In collaborative modeling, modelers and clients work as a team in the modeling process. After completing a set of tasks with respect to a model, a modeler often needs to find out what each of the other modelers has done, or to learn about scenarios run by clients. The research problem suggests that the system should provide a mechanism to display detailed information about how the modeling process has been updated. This involves identifying and sharing changes, highlighting changed components of the modeling process hierarchy, and the freedom to browse the information when required.

1.2 Goals and Motivation

The main goal of this thesis is to provide an infrastructure to capture and support the modeling process hierarchy. The modeling process hierarchy is used to reduce the difficulty in dealing with work directly related to the modeling process.

A second goal of the research is to support the collaborative modeling process by providing change awareness information on top of the modeling process hierarchy. The motivation for this goal is that maintaining change awareness reduces the difficulty in tracking others' changes and in coordinating the progress of the overall modeling process.

1.3 Solution

The research problem states that it is difficult to perform collaborative modeling without explicit support for the modeling process and for change awareness. Our solution is to design a software system that simplifies the bookkeeping tasks in the modeling process by providing an implementation of the modeling process hierarchy for both individual and collaborative modeling. The software system also provides a visual depiction of the modeling process hierarchy, representing either individual or collaborative modeling. The *visual workspace* consists of a set of user interface components that visualize both the modeling process hierarchy and change awareness information. We will discuss how the modeling process hierarchy and visual workspace will aid individual modeling and collaborative modeling below.

The system provides access to both the individual and the collaborative modeling process hierarchies. For the individual modeling process, the system stores all data in an embedded database that is located on the user's computer. For collaborative modeling, the system provides a connection to a remote web server that stores a shared hierarchy. While the connection to a remote model repository is open, the user interaction for operating on a remote model repository is functionally identical to operating on a local model repository. The system allows a user to operate on both local and remote model repositories at the same time. Regardless of the nature of the storage, both mechanisms allow the user to download artifacts from a model repository and execute simulations without the need of the simulation platform provided by modeling software.

The system also provides the user with a visualization and logging infrastructure that helps users track changes within the group. The two change tracking mechanisms are an extension on top of the modeling process hierarchy that shows change information in the visual workspace, and a change list. The visual workspace provides a visual representation of the modeling process hierarchy and highlights changes in the hierarchy through visual cues. The change list collects all changes and displays them in a list-based view.

The system has been designed and implemented to support models built with various modeling software. This allows a broad user base to benefit from the system for both individual modeling and collaborative modeling. Currently supported modeling software includes Anylogic and Vensim. Vensim is modeling software that provides support to build System Dynamics models [11]. Vensim provides an independent dynamic linking library file that allows users to load packaged models and run simulations based on different sets of parameter values. Anylogic is modeling software that supports building System Dynamics models, Agent-based models, and models for discrete event simulations. Support for running a simulation from an Anylogic model is introduced in Chapter 6 [1].

1.4 Steps in the Solution

The following steps have been carried out in completing the solution. Among these, various steps will be implemented in a software tool called SILVER:

(1) Identification of system requirements.

(2) Development of the system infrastructure and support for third party modeling software.

(3) Development of the visual workspace.

(4) Development of a hierarchical representation of the modeling process.

(5) Development of support for the individual and collaborative modeling process.

(6) System evaluation.

1.4.1 Identification of System Requirements

Requirements for the system have been determined in these areas:

(1) The general structure of the modeling process hierarchy.

(2) The type of modeling software needed for constructing and executing models.

(3) The types of change awareness information and features needed for collaborative modeling.

For the first area, the focus was on requirements to provide an access solution for the modeling process hierarchy. The storage solution formulates a virtual structure that consists of modeling projects, model versions, scenarios and related documentation. For the second area, the focus was on supporting external modeling applications (in particular, Vensim and Anylogic) with external support. For the third area, the focus was on features that provide asynchronous change awareness information.

1.4.2 Development of a Modeling Process Hierarchy

The modeling process hierarchy is one area of core functionality provided by the system. This infrastructure needs to provide a technical foundation for data management for both individual and collaborative modeling processes, as well as supporting models from various modeling software. The mechanism to provide such support is implemented in two components.

The implementation of the modeling process hierarchy is based on standard data storage technology. We use databases to provide a storage implementation for the model repository. The storage component is only responsible for maintaining data integrity while transferring and storing data in various modeling contexts. For the individual modeling process, the system creates and manages a single-user model repository on the user's computer. For the distributed collaborative modeling process, the system connects to an existing centralized model repository on a remote database. For the thesis implementation, Java Database Connectivity (JDBC) and Java Transaction API (JTA) were the selected technologies for the repository management middleware. It should be noted that the system design is such that data integrity is maintained in the model repository even if an error occurs during user operation. JTA is a key element making this guarantee possible. Each time a user initiates an operation that affects data storage, a transaction will be created for that operation. If an error occurs during an operation, the associated transaction will cause storage to 'roll back' to the state right before the operation initiated execution.

The data structure and operational logic in the model repository is implemented in the business-logic component, which utilizes the storage component. The purpose of the business-logic component is to support user operations specified in a modeling process hierarchy while delegating technical details of database operations to the storage component. The separation of business-logic and storage will allow a model repository to be hosted on various databases that support JDBC and JTA [46, 47]. This allows us to develop an interface on top of the same set of database operations and supports various modeling software.

This step of the solution produced the architecture and code base for the modeling process hierarchy infrastructure.

1.4.3 Development of the Visual Workspace

The visual workspace is part of the change awareness support used to provide a visual representation of the modeling process structure and associated change information. To build the visual representation, the visual workspace uses the Prefuse toolkit to generate a graph-based visualization and represents each item in the original modeling process hierarchy with a node in the graph [29]. The visual workspace also adds additional visualizations to show useful change awareness information.

1.4.4 Development of Model Package Support

The sections above noted various problems in the modeling process and proposed modeling process elements that should be supported. Because of the limited scope of the project and legal user agreements, complete external support of all modeling software was not developed. Instead, support for Vensim (version 5.6) and Anylogic (version 6.2.2) has been offered. The deliverable from this step in the solution is the suite of modeling software components in the system.

1.5 Evaluation

We asked a group of 5 experienced modelers to perform collaborative modeling using SILVER to evaluate the software. The evaluation process is initial and informal. Answers related the following questions were assessed:

1. Are users able to maintain change awareness using the visual workspace?
2. Does the visualization help the user come to high-level insights from models?
3. What visual cues are most effective for collaboration related tasks?
4. Can the modeling process be improved with support from the visual workspace?
5. How often do users need to interpret the information provided by the visual workspace?

1.6 Contributions

Three contributions are presented in this thesis. The first contribution is a component-based system which implements a modeling process hierarchy to support the modeling process. The second contribution is components – built on top of the first contribution – designed to support change awareness in a collaborative modeling process. The support of the change awareness workspace is discussed in chapter 2, chapter 3 and chapter 5. The third contribution is the evaluation of the system. The results can be found in chapter 7.

1.7 Thesis Outline

The remainder of the thesis is organized as follows:

- Chapter 2 reviews prior work in the area of simulation modeling and groupware. The areas overviewed in the chapter include component-based development, the simulation modeling process and group awareness. The chapter provides background knowledge regarding principles which form the basis of system design.

- Chapter 3 identifies the system support requirements in the distributed modeling process, identifies specific requirements of the change awareness components, and introduces the user interface that is provided in the thesis system implementation.

- Chapter 4 discusses the design and implementation of the modeling process hierarchy infrastructure, underlying storage, and desktop GUI infrastructure of the system.

- Chapter 5 introduces and discusses the visual workspace that was implemented to support change awareness in distributed collaborative modeling.
- Chapter 6 walks through an example of the modeling process conducted with the thesis project.
- Chapter 7 discusses the evaluation of the system, as well as some weaknesses and drawbacks of the system architecture and thesis implementation.
- Chapter 8 provides a summary of the system, discusses the important lessons learned in the course of the research, and proposes directions for future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter provides background information for the research project. Areas of discussion include modeling process steps, modeling methodologies, modeling process hierarchy, collaborative modeling, change awareness and component-based architectures.

2.1 The Modeling Process

The thesis focuses on supporting the process of modeling. This section provides a description of the modeling process and looks at the theoretical foundations of the modeling process as defined by previous researchers in the area of System Dynamics.

2.1.1 Theory of Modeling Process

The modeling process has been described as a scientific process that is iterative, involving a certain amount of trial and error, and often requiring significant time and effort to come to fruition [31]. The nature of iteration reflects the requirement for multiple rounds of revision and evaluation before a model can be further formulated or delivered. With the development of computing power and increased complexity in problems investigated by the modeling process, new modeling methodologies have been developed and supported by modeling software. System Dynamics modeling and Agent-based modeling are two common modeling methodologies.

2.1.2 Overview of System Dynamics Modeling

System Dynamics modeling is a mathematical modeling technique that is used to understand dynamic behaviors and cause-effect relationships among variables in complex systems over time [20]. It aims to capture circular causal effects (feedback) and conceptualize the structure of a complex system through through investigating feedback and accumula-

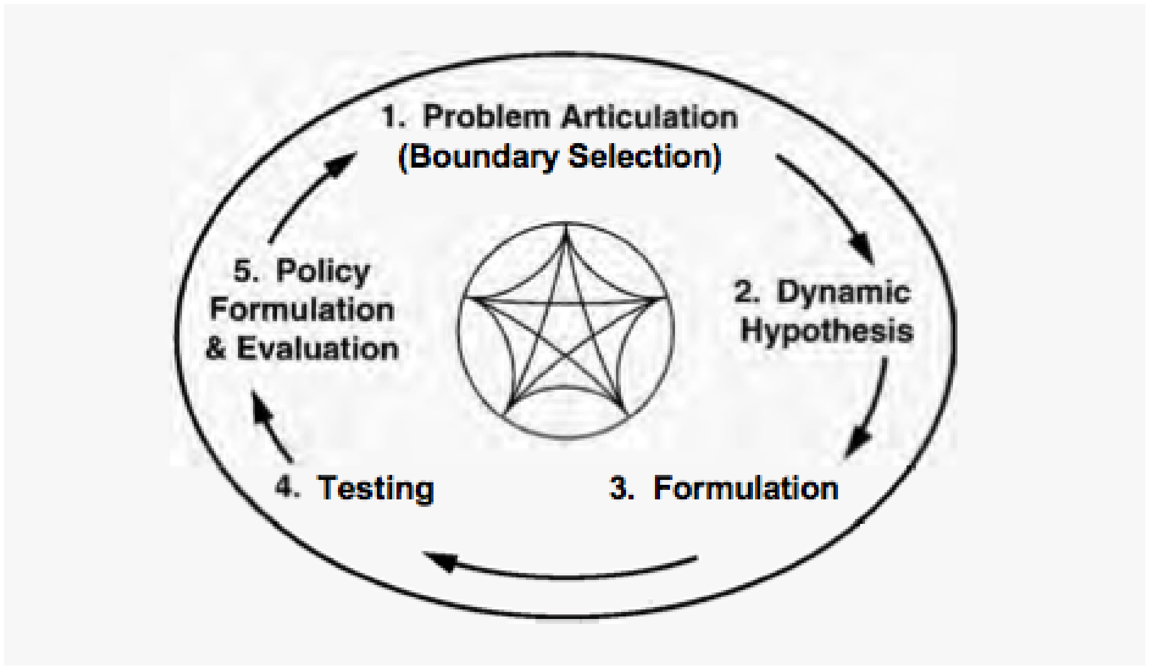


Figure 2.1: Iterative System Dynamics Modeling Process from [55]

tion [20, 21]. Based on Homer's theory of modeling iteration, Sterman described System Dynamics modeling process as an iterative process that involves problem articulation, dynamic hypothesis, formulation, testing, policy formulation and evaluation [55].

In Figure 2.1, most stages in the process commonly involve collaboration between modelers and clients on different subjects. These stages are discussed below:

(a) *Problem Articulation* Problem articulation is a stage where modelers perceive and understand the problem to be addressed. The stage allows modelers and clients to work together to frame the scope of the model by setting a time horizon and variables that are important to understand the problem. The time horizon is the time interval allowed for the symptom of the problem to appear or over which to investigate impact of interventions. Modelers often also need to take notes about background literature and historical data which concerns the domain of the system.

(b) *Formulating a Dynamic Hypothesis* Once the problem has been identified from Part A, modelers must begin to develop a theory, called a *dynamic hypothesis*. A dynamic hypothesis is a working theory of how the problem arose based on the feedback and stock and flow structure of the system. Formulating a dynamic hypothesis requires frequent and close communication between modelers and clients. The goal for such close collaboration is to build a conceptual specification of the model by describing how the system is structured and the rules of interaction. The responsibility of the modeler is to acknowledge and capture details of operations from the clients. The posited hypothesis, underlying the design of the model, is evaluated and tested in the following steps.

(c) *Formulating a Simulation Model* Model formulation is a technical stage where the goal is to let modelers move from a conceptual understanding of a model to a fully specified formal model, which is complete with equations, parameters and initial conditions. The process of formulating a model is technical in nature and generally is centered around modelers operating on modeling software. It may happen that the conceptual model is complex or contains vague ideas that the modeling software is unable to fulfill. The simulation helps modelers identify vague concepts and test their understanding regarding the concept while addressing the uncertainties. To complete a simulation model, a modeler must unambiguously specify model equations, parameters and initial conditions. In larger modeling projects, the model formulation stage may also be accomplished through collaboration by a team of modelers.

(d) *Testing* The testing stage requires real-world historical data or qualitative reference modes applied to the formal model to compare the simulated behavior in the model to the actual behavior of the system. In particular, this stage consists of two important processes: structure verification and sensitivity analysis. Structure verification is conducted by running the simulation with real-world data or reference modes and comparing the state of the model with real-world data or modes for each unit of time. Once the model structure is verified, modelers perform sensitivity analysis by setting initial conditions to alternative (or extreme) cases in terms of both parametric settings and structural settings. The feedback from sensitivity analysis provides modelers and clients with high-level understanding about alternative cases and insights into system behaviours.

2.1.3 Overview of Agent-based Modeling

An Agent-based model (ABM) is a form of computational model where individuals or agents are described as unique and autonomous entities that interact with each other within

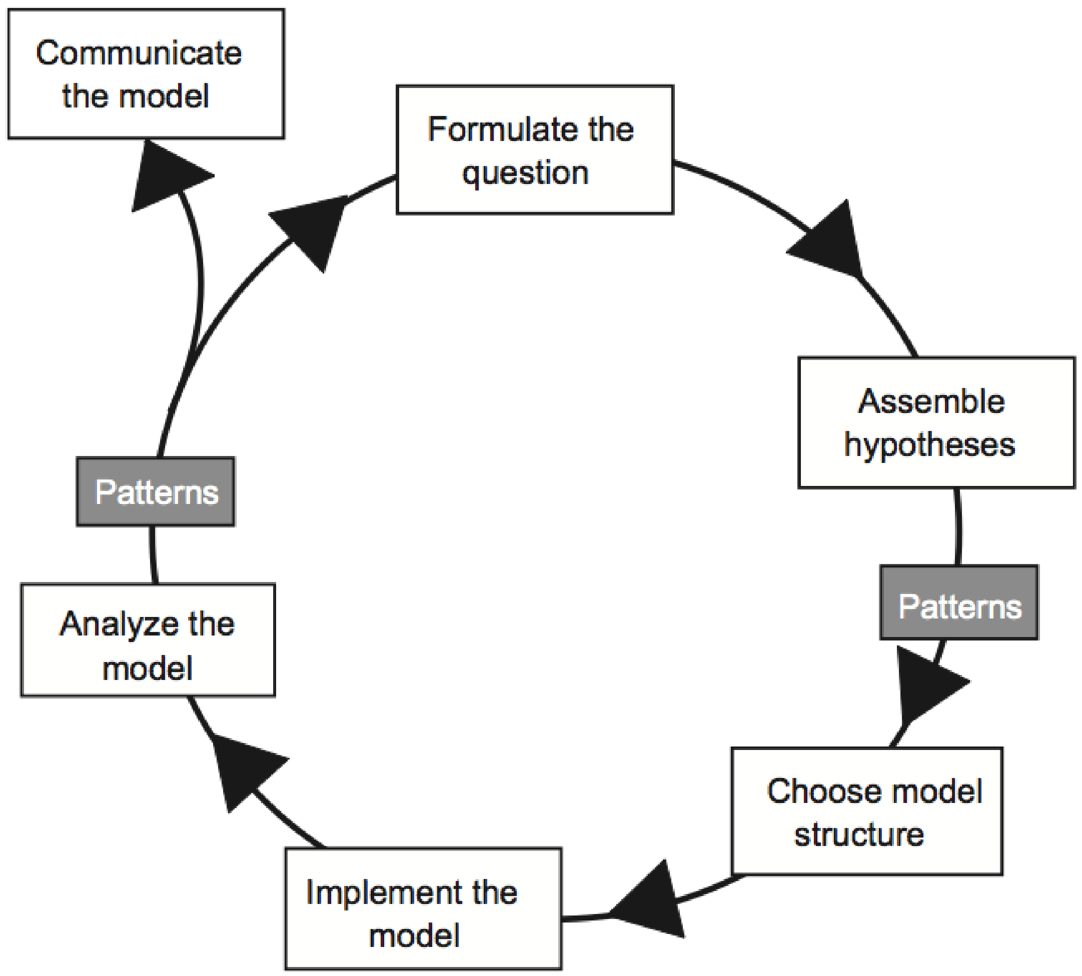


Figure 2.2: The modeling cycle from Railsback et al 2011

an environment [43]. In Agent-based models, agents may be organisms, organizations or any other entity that pursues a certain goal. This modeling technique differs from System Dynamics modeling in several ways, including organizing the model according to individuals in the population (with the state of each such individual stored within the representation for that individual), as opposed to by distinguishing a population according to state (with each state counting the number of individuals who are currently in that state). A similar notion of modeling iteration to that used in System Dynamics is introduced by Railsback et al. [49].

From Figure 2.2, similar modeling tasks are demonstrated for the Agent-based modeling process:

(a) *Formulate the questions.* Similar to the System Dynamics modeling process, this stage suggests starting with clear research questions which will serve as the primary compass and filter for model design. Often this involves identification of patterns that require explanation.

(b) *Assemble hypotheses for essential processes and structures.* The goal for this stage is to identify simplified assumptions and hypotheses to represent the system of interest, then specify agents and their behavior.

(c) *Choose scales, entities, state variables, processes, and parameters.* In this step, based on the hypothesis established above, a written formulation of the model is produced with detailed decisions regarding temporal and spatial scales, entities (different types of elements being modeled, including not just agents but also one or more levels of environments), variables, processes and parameters.

(d) *Implement the model.* This stage is the most technical part of the process, where modelers use modeling software to build a model produced by the written formulation produced by the previous step.

(e) *Analyze, test, and revise the model.* This step involves running the simulation from the established model. The agents' observed behaviours in the simulation will be analyzed to compare to characteristics of real systems (particularly to the patterns identified in step a). Based on the result, modelers and clients will go back to the first step to formulate new questions.

2.1.4 Modeling Process Hierarchy

From the descriptions of the System Dynamics modeling process and the Agent-based modeling process, it is clear that both methodologies adhere to similar steps of an iterative scientific process. While tasks associated with building conceptual models and model formulation differ due to the nature of the applied methodology, the goal for the overall modeling process is to study and understand complex system behavior through an iterative process.

To support a general modeling process that is iterative in nature, Osgood gave definition to a set of terminological conventions associated with the modeling process [48]. A consistent set of terminology was based on the idea of viewing modeling as taking place within the context of modeling projects, with each such modeling project including one or more threads of work that produce incremental versions of modeling artifacts, and associated simulations for each model version. We term the virtual structure that uses these ideas to organize the tasks in the modeling process as the *Modeling Process Hierarchy*. The modeling process hierarchy described by Osgood is comprised of the following parts:

The **Model Project** is the root of the project tree hierarchy which contains all works regarding a model. Each model project will be associated with at least one model version.

The **Model Version** denotes a specific model structure. The initial model project is termed model version one. Future model structural modifications are recorded as model versions in successive order. Each model version may have one or more scenario collections.

A **Scenario** contains assumptions associated with a particular model version. It often specifies values for specific parameters meant to describe some particular "what if" situation. In many pieces of modeling software, output from specific scenario yields a simulation trajectory.

A **Scenario Collection** is a group of one or more scenarios associated with a model version. It is added to ease scenario organizing. Specifically, modelers may group a set of scenarios that share a certain intention in a collection. For example, one scenario collection might contain scenarios used in sensitivity analyses, another might store scenarios representing in investigations of different interventions, yet another those scenarios investigating the impact of assuming different initial state, etc.

2.1.5 Modeling Software

Extant modeling software varies from desktop-based to web-based [53, 5, 8, 11, 1]. The project developed from this thesis provides support for Vensim models as a way to support System Dynamics modeling, and Anylogic as a way to support Agent-based modeling [11, 1].

Anylogic is Java-based software that allows users to build Agent-based models from a graphical user interface and allows users to save the model in a text file. The text file is denoted as an alp file on the computer hard drive and is formatted in XML [38]. Anylogic also allows a user to export a simulation from a particular model to a Java Applet that can be opened and run in a web browser [42].

Vensim is modeling software that allows a user to build System Dynamics models. Version 5.6 comes with the Vensim dynamic-link library which allows us to build third-party applications that interact with Vensim models.

2.2 Awareness

In large group-based collaborative modeling, team-based communication regarding learnings and findings from each stage – and for model formulation, results, evaluation, testing in particular – is important. The importance of ensuring how group meeting and communication is carried out in the modeling process was highlighted in Vennix's group model building theory [58]. Establishing efficient communication in the modeling process is important to success in the modeling process. To achieve this, modelers should be aware of what others in the modeling group are doing. This is awareness about "knowing what is going on" [19]. Human Factors and CSCW researchers have explored and identified six main categories of awareness that a person may maintain:

1) *Situation awareness* concerns people in critical environments. Failure to maintain a minimum level of situation awareness in critical environments may result in undesirable outcomes. In this context, awareness is dependent upon perceiving information in the surrounding environment and interpreting this information so that it is meaningful for the task at hand. Examples can be seen in basketball playing, aircraft piloting and field combat [19].

2) *Informal awareness* supports casual interaction and provides other information such as knowing who is around in the workplace, what others are doing, and whether or not they are available for conversation [17].

3) *Conversational awareness* facilitates how people talk to one another in real time. Awareness is based on visual and audio cues that one picks up when talking to others, and helps one track how the conversation is going [14].

4) *Structural awareness* concerns social and task structures. Structural awareness allows one to know how a work group is organized and the relationships between the people within the group [28].

5) *Workspace awareness* supports people's up-to-the-moment understanding of what is

going on during real-time collaboration in a shared visual workspace. Workspace awareness is limited to events happening in the workspace – inside the temporal and physical bounds of the task that the group is carrying out [28].

6) *Change awareness* is an extension of workspace awareness that focuses on the ability of a person within a group collaboration to track and understand past changes. Change awareness is similar to workspace awareness but differs in focusing on past changes over present events and by supporting asynchronous collaboration rather than real-time collaboration [56].

The collaborative modeling process is concerned with a few categories of awareness listed above. Conversational awareness is required during face-to-face meetings between modelers and clients in problem articulation and model evaluation stages. Considering that most large-scale collaborative modeling processes play out over days to years, change awareness is also required to track past changes in the project. This thesis focuses on using Tam's framework for change awareness to help gather necessary requirements to support change awareness in the modeling process [56].

The framework tries to answer the following main questions regarding change awareness:

- What information makes up change awareness?
- How is change awareness generally supported in collaboration?

The next subsection will provide more backgrounds and details regarding change awareness. Chapter 3 includes further discussion of requirements from collaborative modeling and how change awareness may be supported to fulfill the requirements.

2.2.1 What Information Makes Up Change Awareness?

Change awareness provides information to help answer the question "Is anything different since I last looked at the workspace?" regarding the tracking of past changes in the workspace. Tam's framework for change awareness involves providing answers for who, what, where, when, how and why questions. These categories of awareness for each aspect of change awareness focus on particular informational elements. The categories are shown in Table 2.1 to Table 2.6. For each of the categories, the framework provides information

Where			
Informational elements	Specific Questions		
	Artifact-based view	Person-based view	Workspace view
• Location history • Gaze history • Edit history	• Where was this artifact (when I left)? • Where is the artifact now? • Where has this artifact been during the time that I have been away?	• Where in the workspace has a person visited? • Where in the workspace has a person viewed? • Where in the workspace has a person made changes?	• Where have people been in the workspace? • Where were artifacts in the workspace? • Which parts of the workspace have people viwed? • In which parts of the workspace have people made changes?

Table 2.1: Informational elements related to the 'where' aspect of change awareness [56]

about different views in the workspace. In particular, a person may be looking for:

- Changes associated with an artifact (*artifact-based view*)
- Changes associated with people who made them (*person-based view*)
- Changes within one or a set of locales (*workspace-based view*)

The categories of change awareness combined with location are discussed in more detail in Table 2.1.

The informational elements pertaining to 'where' questions include location history, gaze history and edit history. Location history describes parts of the workspace that a person has visited. Gaze history describes parts of the workspace a person has looked at. Edit history describes parts of the workspace in which people have made changes. The 'who' elements are presence history, identity, readership history and authorship history. Presence history describes who has made changes in the workspace. Identity identifies all specific individuals associated with a change or an event. Readership history lists

Who			
Informational elements	Specific Questions		
	Artifact-based view	Person-based view	Workspace view
• Presence history • Identity • Readership history • Authorship history	• Who has looked at this artifact? • Who has changed this artifact?	• Who has this person interacted with? • Who made changes with this person?	• Who has been in the workspace? • Who has looked at the workspace? • Who has made changes to the workspace?

Table 2.2: Informational elements related to the 'who' aspect of change awareness [56]

What			
Informational elements	Specific Questions		
	Artifact-based view	Person-based view	Workspace view
• Action history	• What changes have been made to the artifact?	• What artifacts has a person looked at? • What artifacts has a person changed? • What activities has a person engaged in?	• What changes have occurred in the workspace? • What artifacts were viewed? • What artifacts were changed?

Table 2.3: Informational elements related to the 'what' aspect of change awareness [56]

How			
Informational elements	Specific Questions		
	Artifact-based view	Person-based view	Workspace view
• Process history • Outcome history	• How has this artifact changed?	• How has a person changed things?	• How has this workspace changed?

Table 2.4: Informational elements related to the 'how' aspect of change awareness [56]

When			
Informational elements	Specific Questions		
	Artifact-based view	Person-based view	Workspace view
• Event history	• When was this artifact changed? • When was a particular change to this artifact made? • In what order were changes made to this artifact?	• When did a person make changes? • When did a person make a particular change? • In what order did this person make changes?	• When were changes made to the workspace? • When did a particular change in the workspace occur? • In what order did changes to the workspace occur?

Table 2.5: Informational elements related to the 'when' aspect of change awareness [56]

Why			
Informational elements	Specific Questions		
	Artifact-based view	Person-based view	Workspace view
• Cognitive history • Motivation history	• Why was this artifact changed?	• Why did a person make that change?	• Why was that change made in the workspace?

Table 2.6: Informational elements related to the 'why' aspect of change awareness [56]

all the people who have viewed a particular artifact, or, conversely, listing all the items that a particular person has seen. Similar to readership history, authorship history lists the people who have made changes to an artifact. The 'what' element is action history. Action history varies depending upon the perspective (or "view") that a person has of the workspace. With the artifact-based view, action history describes changes that have made to a particular artifact. With the person-based view, action history describes what actions a person has undertaken. With the workspace-based view, the questions ask about the actions that were carried out on the artifact in the workspace. The 'how' elements are process history and outcome history. Process history describes how the project evolved from a previous state (e.g., the state that it was in when a person last looked at it) to its present state. Outcome history describes the 'bottom line' understanding of a change by only reporting those things that differ between the initial and the final state. The 'when' element is event history. Event history provides chronological context for understanding and interpreting changes. The 'why' category reflects the need for knowing the thought and motives behind the change. The two elements in the 'why' category are cognitive history and motivational history. Cognitive history describes the logic or reasoning that may be behind a change. Motivational history deals more with the desires that are the impetus for making a change.

2.2.2 How is Change Awareness Generally Supported in Collaboration?

Change awareness is widely used in asynchronous collaboration, and change awareness information can be supported in several ways:

Version Control automates and enhances the process of manually tracking progressive versions of documents and their changes [37].

File differencing occurs when a user manually specifies two documents and asks the system to compare them for differences [32].

Overviews communicate changes made to an entire artifact at a glance [30].

Highlighting displays differences within the artifact itself by directly marking up the appearance of the artifact structure or contents [12].

History systems track all incremental changes made to a document, typically so they can be played back at a later time or so actions can be undone [35].

One of the main focus of the system contributed by this thesis is to provide necessary change awareness information to users. In particular, the system focuses on providing information for the categories of ’where’, ’who’, ’what’ and ’when’. In order to do so, information related to changes in the modeling process hierarchy must be tracked.

2.3 Component Software

According to Szyperski, component software is a coherent application comprised of separate pieces called components. Sparling provides a more concise definition for a component: “A component is a language neutral, independently implemented package of software services, delivered in an encapsulated and replaceable container, accessed via one or more published interfaces.” (p. 47) [15, 54].

According to Sparling, component software provides a few advantages over traditional software design. First, it allows component to be reused without knowing the implementation details of the component. Next, it lowers development costs and shortens development timelines. Also, components can be plugged and unplugged since they are separate and independent. Finally, component version upgrades typically do not require changes to applications due to well-defined public interfaces. The work of this thesis involves the creation of a system that facilitates the modeling process by assembling components that support various modeling software. As a result, the thesis work will benefit from use of a component software platform. Equinox OSGi has been selected as the component platform to implement the work for this thesis and is discussed in the next subsection.

2.3.1 Equinox OSGi

The system contributed by this thesis is a component software package built with Equinox OSGi [57, 40]. *OSGi* stands for Open Services Gateway initiative. It is a module system and service platform for the Java programming language that implements a complete and dynamic component model. This section provides an introduction to the Equinox OSGi specification and features of the OSGi platform. Equinox OSGi is an implementation of the OSGi specification [6]. As an implementation framework, Equinox OSGi provides an environment for the modularization of applications into smaller bundles such as a GUI bundle, a database bundle and a middle-ware bundle. Each bundle is a tightly coupled, dynamically loadable collection of Java classes, Java jar files and configuration files used to explicitly declare external dependencies. Equinox OSGi has undergone heavy testing and is robust enough to provide a program runtime for the popular Eclipse IDE [23].

CHAPTER 3

SYSTEM REQUIREMENTS

This chapter discusses the first step of the thesis solution: gathering requirements for system support for collaborative modeling. The requirements analysis involves the following activities:

(a) Identify and determine the basic operations associated with the modeling process hierarchy that can be used to support both individual and collaborative modeling processes.

(b) Determine the user interface (UI) for the system to allow users to participate in both individual and collaborative modeling processes.

(c) Identify the information and changes that are to be tracked by the system.

(d) Determine the strategy for integrating a collaborative workspace into SILVER to support change awareness.

3.1 System Integration for the Modeling Process Hierarchy

The idea of Osgood's modeling process hierarchy, or MPH, has been introduced in section 2.1.4. It provides a virtual structure to help organize modeling process iteration. While Osgood's MPH suggested Model Project as the highest level (root) of the structure, we added Group and User to the structure to address requirements associated with the collaborative modeling process. In the new MPH structure, *User* is the highest level of the hierarchy for a user. Each 'User' item contains one or multiple 'Group' items to specify the creation and ownership relationship. A *Group* item represents a virtual work group that involves multiple model projects. From the user's point of view, each item in the MPH represents an entity or a step in the modeling process.

SILVER provides an implementation of the MPH and a set of available user actions for each type of item in the MPH. To achieve this, we first identified a list of actions that apply to the needs of one or more types of MPH items:

(1) Create and add an item.

(2) Import information from an external model and create a corresponding item.

(3) Update fields in a particular item, such as item name, description, etc.

(4) Duplicate an item, including all included items.

(5) Remove an item.

(6) Associate an external file with this item.

(7) Run a simulation with a given parameter set (a mapping from parameters to values).

In Table 3.1, the labeled actions above are mapped to items types in the MPH. The table is used to indicate the available user operations on an item in SILVER.

Modeling Process Hierarchy Items	**Supported Actions**
Group	1, 3, 5,6
Model Project	1, 2, 3, 5, 6
Model Version	1, 2, 3, 5, 6
Scenario Collection	1, 2 (Vensim only), 3, 5, 6, 4
Scenario	1, 2, 3, 5, 6, 4

Table 3.1: Supported Actions in MPH

3.2 Modeling Process Support

The previous section introduced available user operations in MPH. This section will explain how SILVER is a solution to many gaps in practice in the modeling process [48]. A few gaps that were mentioned by Osgood are mentioned below:

3.2.1 Modeling Version Control

Version control is a technique widely used to support the software engineering processes that allows people to keep multiple snapshots of an artifact and to be able to roll back to previous snapshots if required [37]. However, the version control technique is uncommon in the modeling domain. As pointed out by Osgood, lack of the version control technique can easily result in an inability to retrieve earlier versions of the model. In SILVER, The structure of the MPH should allow a user to create a new model version by importing a model. As a result, this technique allows saving snapshots of a model after a structural change (e.g. adding a variable) is made by modeling software.

3.2.2 Scenario Bookkeeping

An important relationship exist between version, parameters and simulations simulated. Such relationship is termed scenario by Osgood [48]. Often, scenario bookkeeping are not directly supported by modeling software, thus a tedious to maintain. In SILVER, MPH should support scenario bookkeeping by giving users the ability to run model simulations and by saving results in each version of the model. Simulation results include captured datasets with certain parameters and simulation screenshots. Scenario bookkeeping also strongly benefits from the ability to be able to record assumptions by updating parameter values, re-running the simulation and saving the results.

3.2.3 Linking External Artifacts

In many situations, external files might be needed as a reference in explaining or motivating decisions that are made during the modeling process, or in further exploring model results. As the modeling project progresses, the amount of external files grows easily, which makes it difficult for modelers to record where in the modeling project the external file should be associated with. An implementation of MPH should allow users to attach external files to items at multiple levels of the modeling hierarchy.

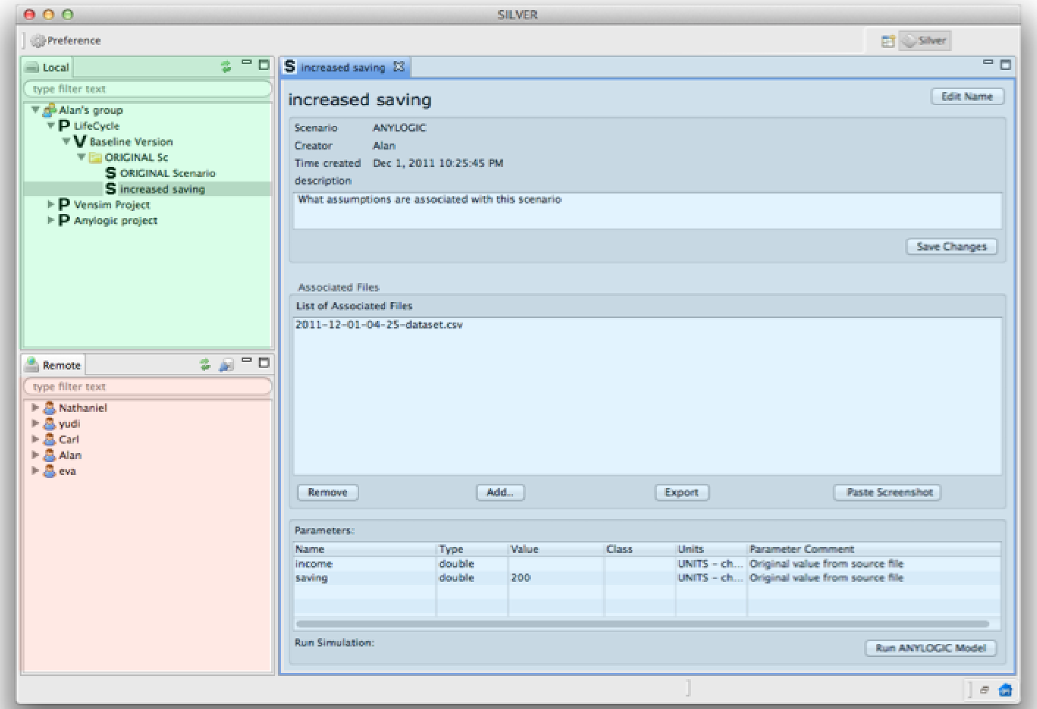
3.2.4 Simulation Support

While the integrity of information in model building is important, the relationship between the applied assumptions and corresponding simulation outputs is the key to successful model analysis. Modelers have no support from the modeling software but have to remember maps to correlate model outputs and corresponding assumptions within a set of past simulations. Simulation support should allow users to check simulation results or re-run the simulation from the corresponding scenario that records assumptions.

3.3 User Interface In SILVER

3.3.1 Main Interface

The interface for SILVER has been designed based on the requirements gathered from section 3.1.



increased saving

Scenario	ANYLOGIC
Creator	Alan
Time created	Dec 1, 2011 10:25:45 PM
description	What assumptions are associated with this scenario

Associated Files

List of Associated Files

2011-12-01-04-25-dataset.csv

Parameters:

Name	Type	Value	Class	Units	Parameter Comment
income	double			UNITS – ch...	Original value from source file
saving	double	200		UNITS – ch...	Original value from source file

Run Simulation:

Figure 3.1: Overview of the SILVER User Interface

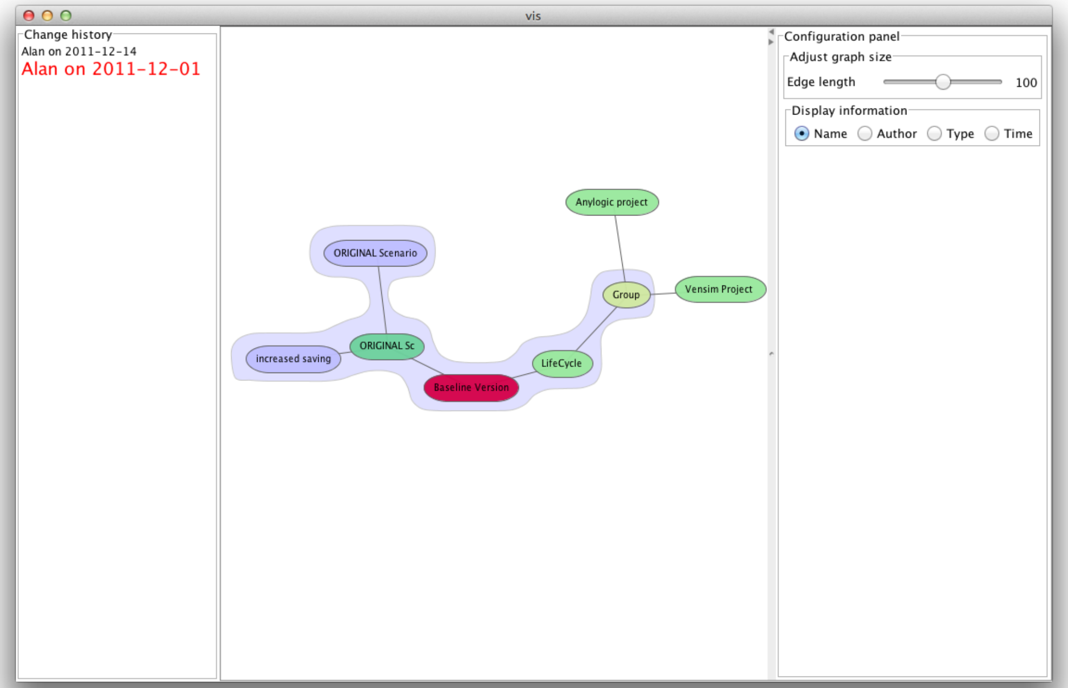
Figure 3.1 displays the main user interface of SILVER. Highlighted by three different colors are three components associated with distinct occupied display regions in the main interface. The green area (top left panel) represents the MPH view that is physically stored on the user's computer and is to be used in the individual modeling process; we name it the *Local MPH View*. The red area (lower left paanel) represents the MPH view that is physically stored on a remote web server and is accessed through Internet; we name it the *Remote MPH View*. The blue area (right side of screen) represents detailed information about selected items in the MPH views; we name it the *Information Panel Area*. Within the information panel area, the user can edit and update entries about an item such as its name or description. Once the main user interface is present, a user will typically start browsing or creating items in either the local MPH view or the remote MPH view. Also, the user is able to work with both MPH views simultaneously (e.g. dragging a project from the local to the remote area).

3.3.2 Dedicated Collaborative Workspace for Change Awareness

SILVER supports change awareness by offering users two components: a visual workspace and a change history panel. Figure 3.2 is the visual workspace component in SILVER. It can be activated either from a group item or a model project item. Based on the item selected, a visual workspace component interprets all MPH items that are associated under the item and constructs a visual workspace in a separate window. The visual workspace comprises three column views.

The left column displays the change history associated with the displayed items, using a fish-eye visualization [25]. The change history responds to mouse hovering by focusing and zooming in on the item with high magnification and detail. The fish-eye visualization technique is used to display a history of changes primarily because people are interested in seeing the detail of selected parts without losing overall context.

The middle column displays an animated graph using a force-directed layout algorithm [24]. As part of the graph, each node represents a particular item in the MPH; all nodes are connected based on their relationship in the particular MPH being represented. Double-clicking on the node will show information about the corresponding item in the main user interface. The force-directed algorithm grants a graph the ability to self-balance by adjusting edge lengths based on the number of connections of the node until an overall state of balance has been reached. The graph-based MPH responds to a change in the selected date from the change history in the left column by using a Bubble Set visualization to include the nodes which represent the items that have been changed on that date [16]. As shown in Figure 3.3, six items that have been changed on the selected date of December 1st are


vis

Change history
Alan on 2011-12-14
Alan on 2011-12-01

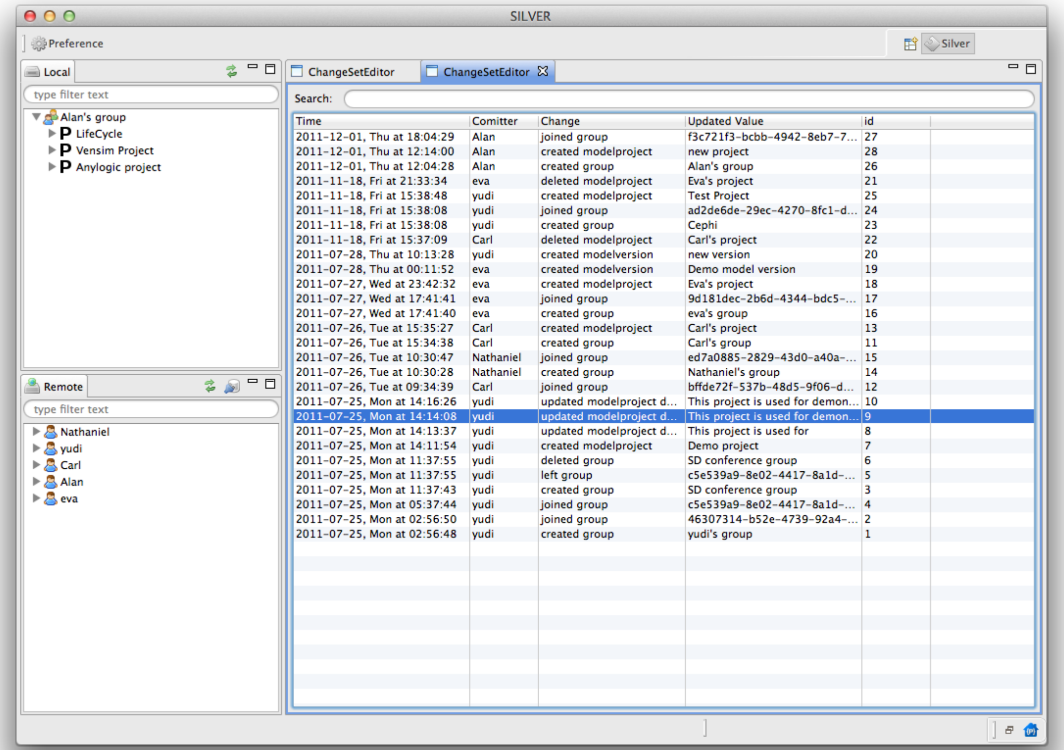
Configuration panel
Adjust graph size
Edge length 100
Display information
[✓ Name] Author Type Time

Figure 3.2: SILVER Visual Workspace

highlighted. Selecting other dates will result in different sets of nodes being highlighted to reflect locations of changes.

The right column contains a set of control groups, which are the "configuration panel" group and the "display information" group. The "configuration panel" group holds a slider which will adjust the length of the edge. It is used to adjust the overall size of the graph to the user's most comfortable view and to reduce clutter in display area. The "display information" group displays a set of radio buttons which are "Name", "Author", "Type" and "Time". Each radio button represents content to be displayed in the nodes of the MPH graph visualization.

Figure 3.3 shows the change history panel that can be activated through summoning a context menu in the MPH and choosing "Show list of changes." The change history contains the full log of changes in a list view, regardless of the particular item on which the change occured. Each row record listed in the change log represents a change that was made by a user at a specific time. Except for the "remove" change, double-clicking a record in change log will display detailed information of the particular MPH item that has



Time	Comitter	Change	Updated Value	id
2011-12-01, Thu at 18:04:29	Alan	joined group	f3c721f3-bcbb-4942-8eb7-7...	27
2011-12-01, Thu at 12:14:00	Alan	created modelproject	new project	28
2011-12-01, Thu at 12:04:28	Alan	created group	Alan's group	26
2011-11-18, Fri at 21:33:34	eva	deleted modelproject	Eva's project	21
2011-11-18, Fri at 15:38:48	yudi	created modelproject	Test Project	25
2011-11-18, Fri at 15:38:08	yudi	joined group	ad2de6de-29ec-4270-8fc1-d...	24
2011-11-18, Fri at 15:38:08	yudi	created group	Cephi	23
2011-11-18, Fri at 15:37:09	Carl	deleted modelproject	Carl's project	22
2011-07-28, Thu at 10:13:28	yudi	created modelversion	new version	20
2011-07-28, Thu at 00:11:52	eva	created modelversion	Demo model version	19
2011-07-27, Wed at 23:42:32	eva	created modelproject	Eva's project	18
2011-07-27, Wed at 17:41:41	eva	joined group	9d181dec-2b6d-4344-bdc5-...	17
2011-07-27, Wed at 17:41:40	eva	created group	eva's group	16
2011-07-26, Tue at 15:35:27	Carl	created modelproject	Carl's project	13
2011-07-26, Tue at 15:34:38	Carl	created group	Carl's group	11
2011-07-26, Tue at 10:30:47	Nathaniel	joined group	ed7a0885-2829-43d0-a40a-...	15
2011-07-26, Tue at 10:30:28	Nathaniel	created group	Nathaniel's group	14
2011-07-26, Tue at 09:34:39	Carl	joined group	bffde72f-537b-48d5-9f06-d...	12
2011-07-25, Mon at 14:16:26	yudi	updated modelproject d...	This project is used for demon...	10
2011-07-25, Mon at 14:14:08	yudi	updated modelproject d...	This project is used for demon...	9
2011-07-25, Mon at 14:13:37	yudi	updated modelproject d...	This project is used for	8
2011-07-25, Mon at 14:11:54	yudi	created modelproject	Demo project	7
2011-07-25, Mon at 11:37:55	yudi	deleted group	SD conference group	6
2011-07-25, Mon at 11:37:55	yudi	left group	c5e539a9-8e02-4417-8a1d-...	5
2011-07-25, Mon at 11:37:43	yudi	created group	SD conference group	3
2011-07-25, Mon at 05:37:44	yudi	joined group	c5e539a9-8e02-4417-8a1d-...	4
2011-07-25, Mon at 02:56:50	yudi	joined group	46307314-b52e-4739-92a4-...	2
2011-07-25, Mon at 02:56:48	yudi	created group	yudi's group	1

Figure 3.3: SILVER Change History Panel

been changed. The list of changes contains five columns of data: "Time", "Committer", "Change", "Updated Value" and "id". The display of changes can be sorted based on each column category. The "Time" column specifies the time of change made; the "Committer" column indicates the name of the user who made the change; the "Change" column gives a description of the change; the "Updated Value" column specifies the new value that resulted from the change; "id" is an incremental integer value which is used to help sort according to the temporal order of the changes.

3.4 Change Awareness Information in Each User Interface Component

For each type of user interface component identified above, the following sections list a table of the features the system supports for each change awareness category.

(a) *MPH Views*

Both local and remote MPH views do not provide awareness features themselves.

(b) *Information Panel*

Table 3.2 shows change awareness support from the information panels in SILVER.

Information Panel		
Category	Information	Support Features
Who	Identity	The name of creator is shown in the item's information panel.

Table 3.2: Supported change awareness in the information panel

(c) ***Visual Workspace***

Table 3.3 shows the change awareness support of the visual workspace in SILVER.

(d) ***Change Log Panel***.

Table 3.4 shows change awareness support from change log panels in SILVER.

The software requrements addressed in SILVER demands support to diversified features and configuration. The next section will introduce SILVER infrastructure as a solution based on component-based software architecture.

Visual Workspace		
Category	Information	Support Features
Where	Location history	Location of changes are highlighted in the form of Bubble Set visualizations on the MPH graph.
	Edit history	What MPH item (e.g., Model Version) has been changed in the past?
Who	Identity	The identity of the person that has made a change can be found in the change history of the visual workspace.
	Authorship history	Name of people who have been involved in the project can be seen by switching “Display information” to “Author” in the visual workspace.
What	Action history	Partially supported by selecting a date from the change history view; the changed items will be highlighted in the visual history.
When	Event history	Supported by change history view: time of changes are listed in history view.

Table 3.3: Supported change awareness in the visual workspace

Change Log Panel		
Category	Information	Support Features
Where	Edit history	Edit history can be seen by showing change logs in chronological order.
Who	Authorship history	Authorship history for a particular item can be seen by showing change logs ordered by item names.
What	Action history	Partially supported by either re-ordering change logs by “Committer” or by “Time”
When	Event history	Supported by re-ordering by “Time” column in the change log.

Table 3.4: Supported change awareness in the change log panel

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

This chapter discusses the second and third steps in the thesis solution introduced in Chapter 1: development of the modeling process hierarchy infrastructure, and development of the graphic user interface (GUI) infrastructure. These steps comprise the design and implementation phases for the core of the system.

SILVER is a multi-layer architecture, as shown in Figure 4.1 below. Each rectangular component is modular and hides the implementation details. At the base of the SILVER architecture is the Eclipse OSGi and Eclipse Runtime platform. The OSGi platform, mentioned in Section 2.3.1, reduced the development effort in specifying dependencies between components and shortened the development process by providing pre-built components. The storage component is responsible for providing a database storage mechanism to construct and modify the structure of an MPH either on a user's computer or on a web server. The business logic component is built on top of the storage component and provides support that helps allow both front-end GUI components and visual workspace components to access and modify the modeling process hierarchy. The GUI component encapsulates all user interface and interaction related entities and is responsible for the main user interface. The visual workspace is responsible for providing the user with a set of visualizations as an alternative way to browse the modeling process hierarchy. Both the GUI component and the visual workspace component use the business logic component to modify information related to the modeling process hierarchy.

This chapter gives a high-level overview of the highlights and features of SILVER, which is followed by a discussion of the design and implementation of the storage component infrastructure, business logic component infrastructure, GUI component infrastructure, and the visual workspace infrastructure.

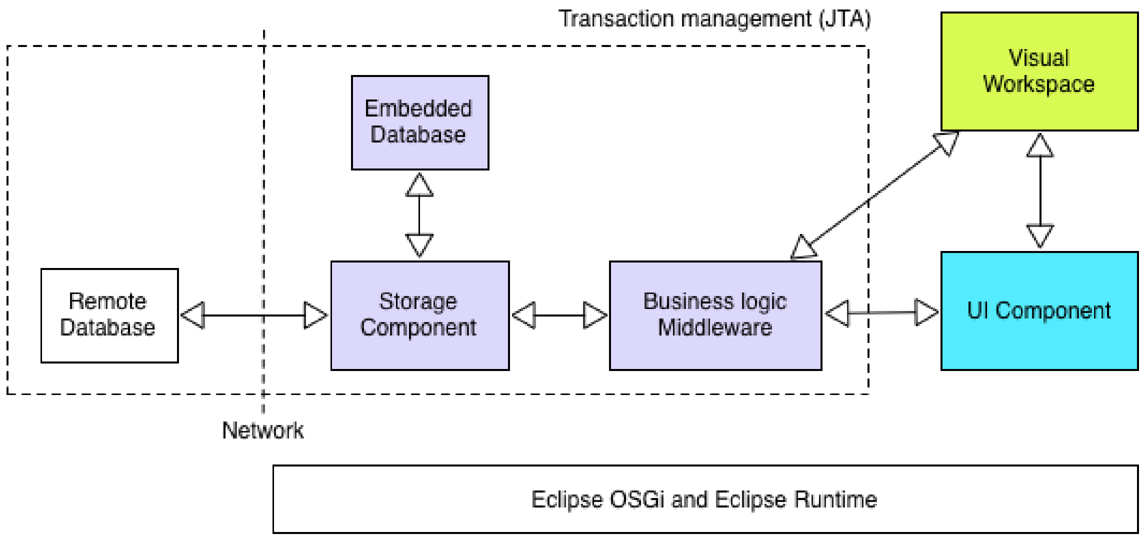

Transaction management (JTA)
Visual Workspace
Embedded Database
Remote Database
Storage Component
Business logic Middleware
UI Component
Network
Eclipse OSGi and Eclipse Runtime

Figure 4.1: A High-Level View of the SILVER Architecture

4.1 New Support Features Provided by SILVER

The SILVER system attempts to fill several gaps in the modeling process in terms of collaboration, which existing modeling software fails to support. These gaps are caused by a lack of a generic and extensible visual workspace that provide workspace awareness support. Due to the scope of this project, SILVER will not include a complete implementation for all workspace awareness features of the system that were listed in Chapter 3. SILVER provides a component-based architecture with an API that will allow expansion of the system for further functionality. The purpose of developing the system is to demonstrate that the proposed architecture fulfills basic user requirements for MPH that facilitate change awareness. The significant and unique features that the proposed system provides are listed below:

1. A fully extensible modeling process hierarchy infrastructure that provides support to basic model version control, attaching external documents and scenario bookkeeping.

2. A concise visual workspace that provides change awareness to help user track changes in the collaborative modeling process.

3. Support for two popular pieces of external modeling software by directly running simulations and capturing simulation output and stopping conditions.

4.1.1 System Highlights

This section gives a high-level overview of the architecture and design of the system, as well as of the significant features that have been built into the system. The system has been developed using the following technologies:

1. *Java programming language.* Java's strong object-oriented nature and platform independence provides SILVER with sufficient technology support. Additionally, Java as a platform is popular and widely-used for various toolkits, include those that were used to build SILVER.

2. *Eclipse Equinox OSGi and RCP.* Both Equinox and RCP (Rich Client Platform) are Eclipse open-source projects that are essential parts of the Eclipse integrated development environment (IDE). Equinox provides implementation to the OSGi bundle specification. The term "bundle" is equivalent to the meaning of "component" in component-based architectures. Equinox OSGi forms the basis of the Eclipse Runtime. RCP is a set of integrated tools that allow developers to build desktop software with OSGi based components. This allows SILVER to be built upon the Eclipse runtime and to integrate other components that, in turn, allow it to be integrated as an OSGi bundle.

4.1.2 Distribution Architecture

In terms of distribution architecture, the system provides:

1. An embedded database storage component that allows a user to create and maintain a model repository in SILVER on the modeler's computer. The created model repository can be accessed/modified without an Internet connection.

2. A remote database storage server used by SILVER when users plan to share or collaborate over the Internet. Since the server component is not the main focus of the system, the choice of distribution management was delegated to the database.

In terms of distributed communication and message passing, the system uses:

1. *Java Transaction API.* JTA is a high-level specification in Java that allows SILVER to perform distributed transactions that access and update data on two or more net-

worked computer resources. The system makes use of an open source transaction manager to process transactions at the application level. *Transaction Processing* allows multiple individual operations to be linked together automatically as a single, indivisible transaction. The transaction-processing system ensures that either all operations in a transaction are completed without error, or none of them are [27]. In SILVER, once a user issues a command to retrieve or modify data, a transaction will be initiated to carry out one or more tasks to achieve the goal. If a task fails due to an unexpected error during execution, the data that were to be modified by that transaction will remain intact. Applying JTA will prevent data stored in SILVER from being corrupted due to unexpected error conditions. This is particularly important when modelers are collaborating through an unreliable or slow Internet connection.

2. *JDBC Connections*. SILVER uses Java database connectivity, or JDBC, to maintain connections to both local and remote database at the same time [46]. This allows the user to work with two model repositories stored at two different locations (one on the local computer, the other at the remote host) simultaneously. All data that are passed over JDBC will be managed by an application-level transaction manager that provides JTA support.

In terms of the user interface and change awareness support, the system:

1. Provides change awareness information processing for data stored on both embedded and remote storage;

2. Displays the model process hierarchy visualization with awareness support in an interactive visual workspace.

In terms of external modeling software support, the system supports:

1. Anylogic logging. SILVER supports Anylogic simulation logging by applying AspectJ load-time weaving to Anylogic exported simulations [50]. Once the simulation is finished, the values of DataSet objects will be logged to csv files and attached to the corresponding scenario in SILVER. In addition, the simulation result page will be copied into a snapshot picture and attached to the same Scenario in SILVER.

2. Vensim logging. SILVER supports Vensim logging by using the Vensim DLL provided by Ventana System Inc [11]. The Vensim DLL only works in the Windows environment due to limitations on cross-platform DLL support.

4.2 System Infrastructure

At the heart of SILVER is the storage component. This component is designed to hide all the technical details about database related operations from the rest of the system by providing a generic programming interface. In particular, the storage component has two responsibilities. The first responsibility is to establish connections between SILVER and databases on hard drives or over the internet; the second responsibility is transactional management, which maintains data integrity. It also provides implementation to granular and basic operations to create and modify the model repository for databases.

On top of the storage component is the business logic component. The purpose of the business logic component is to make use of the storage component to provide support to build model repositories for various modeling applications. The business logic component is designed with stateless Java objects to ensure data transactionality and scalability to large-scale concurrent processing. It is also to ensure that the system provides guarantees (part of the ACID transactional guarantees) by using the data storage mechanisms.

4.2.1 General Design Goals

For the scope of the system implementation of SILVER, there were a number of design goals. These goals are outlined below:

1. *Lightweight data traffic.* The data traffic between SILVER and a remote storage server needs to be optimized for repetitive or batch operations.

2. *Generic and flexible design.* The database storage schema should be flexible and generic enough to be used by a "black box" storage implementation with a number of relational databases that support JDBC.

3. *Extensibility.* The design should be extensible such that new types of models and simulations can be created and incorporated into the infrastructure, supporting of new modeling software or other aspects of change awareness.

4. *Robustness and Stability.* Regardless of the system configuration, the system ought to be robust and stable in daily modeling operations.

5. *Data integrity.* The design should maintain data integrity and prevent it from being corrupted by unexpected errors.

4.2.2 System Implementation Goals

For the thesis implementation of the system architecture, there are some extra goals based on the technology used for the implementation. More specifically, there are goals of the implementation based on development using the Java language. These goals are outlined below.

1. *Use of Transaction Management.* The Java JDBC protocol includes transaction management on a per database connection basis. A *Connection* is a request to the database that generally modifies stored data or finds information based on a SQL query. By default, each connection will initiate a transaction before contacting the database server to ensure the restore point is set up for that particular session. In SILVER, transaction management has been made to support nested transactions, where a transaction may be composited of multiple connection requests to a database. Such support is mainly used in the business logic component, which initiates multiple requests in scenarios, such as importing and recreating a model project from an existing model.

2. *Use of Object Pooling.* Object pooling is a technique to create a pool of Java objects before any execution in the system. The goal of the technique is to enhance scalability by leveraging the cost of initializing an object at runtime and reduce memory use overall [41]. In SILVER, the technique is used in the business logic component to initialize a pool of stateless objects.

3. *Use of XML Data Representation.* XML is a language for defining markup languages [38]. XML allows information to be more usable and accessible. In SILVER, the structure of the entirety or part of the modeling process hierarchy are represented in XML format while being transferred between the storage component and the business logic component.

4.2.3 Business Logic Component Design

The MPH concepts are important to coordinate collaborative modeling work flow. The business logic component is middle-ware that encapsulates the logic defined in the MPH and determines how support for multiple modeling software can be integrated into the process. In particular, the business logic component is responsible for maintaining the specification, implementation and external modeling package support for the MPH. The requests to retrieve or modify information are passed from the client component to the business logic component. Based on the context of the request, the business logic component initiates a transaction and passes the request to the storage component. As mentioned in the previous section, a transaction ensures that one or more actions are executed in a unit; either all actions are executed, or none of the actions is executed. In the example shown in Figure 4.2, the flow of requests within the business logic component is conducted in six steps:

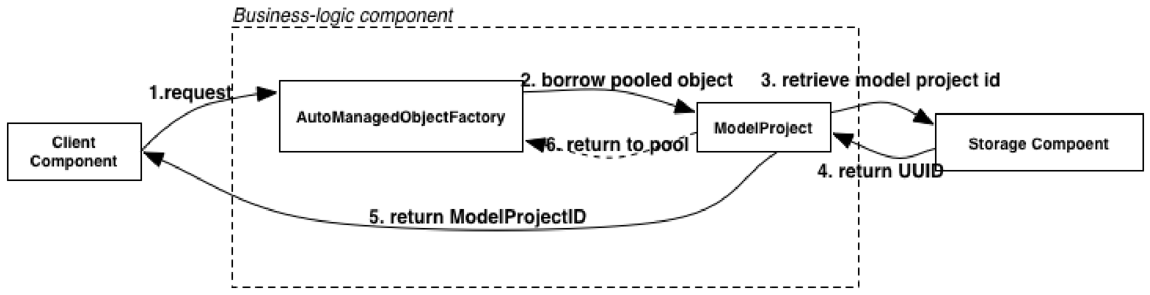


Figure 4.2: Example request flow in the business logic component

(Step 1) A request is initiated by a user action from the client side, and originates either in the desktop software or the visual workspace. The request in the example scenario here is to create a model project item within MPH.

(Step 2) The AutoManagedObjectFactory provides a (stateless) ModelProject object from the ModelProject object pool to carry out the request.

(Step 3) The ModelProject object initiates a new transaction and calls the corresponding service from the storage component to create a new model project.

(Step 4) The storage component evaluates the information from the request, creates a new model project and passes back an a universally unique identifier, or UUID object that represents the newly created data back to the ModelProject object.

(Step 5) The ModelProject object further creates a ModelProjectID based on the id data UUID and passes it back to the Client Component.

(Step 6) After step 6 is finished, the ModelProject object is returned to the ModelProject object pool for future use.

The MPH interfaces inside the business logic component have been implemented using the object pooling pattern [41]. Because the MPH implementation follows the pooling pattern, the class diagram in Figure 4.3 shows the general MPH and related classes of the business logic component.

At the left of the diagram is the MPH programming interface, found in the stateless package. The MPH programming interface consists of 7 interfaces. Each interface is implemented by a concrete class designed as stateless object. A stateless object is simply a group of procedures and does not retain observable state at application runtime. Except for the SilverService interface, all other interfaces are provided so as to support each item in the modeling hierarchy. The SilverService interface provides support to configuration details related to the database and to external modeling software. The MPH-related interfaces include GroupI, ModelProjectI, ModelVersionI, UserI, ScenarioCollectionI and ScenarioI. Each of them specifies operations that relate to the MPH. Each interface is followed by a default implementation class, which is labeled with a name similar to that of its interface. Instead of having client classes directly instantiate objects, all implementation objects are managed by a pooling object created by the AutoManagedObjectFactory class. As mentioned in Step 2 above, the object pool creates a set of objects that can be reused anywhere in the system. It bears noting that different simultaneous uses of the business logic (as might be required to support a web GUI or a web service) must use separate objects from the pool, so that they make use of different connections (each associated with different transaction state). Once a designated task completes its use of the object, the object will return to the pool and wait to be used later. Depending on the infrastructure, there are many object pooling techniques available. SILVER uses reference object pools provided by the Apache Commons Pool library, which will activate garbage collection to reduce the JVM's heap if necessary [2, 36].

At the right of the Figure 4.3, the TransactionalManager class provides transaction management for all stateless objects that provide implementation to the interfaces. As illustrated from Step 3 to Step 5 in Figure 4.2, a transaction is created to include all operations, including those that are performed in the storage component. The transaction will recognize and manage all new transactions that are created in the storage component. The transaction manager will ensure no operations are performed unless no error has been raised from the beginning of the operation until the end of the operation. As illustrated by Step 5, all transactions that represent a sub-task of the operation, or parent transaction, will be committed together to perform the modification. This mechanism is very common in many enterprise applications and web applications that involves relational database support. It is extended to the application level in SILVER to prevent database corruption due to errors between a set of combined operations.

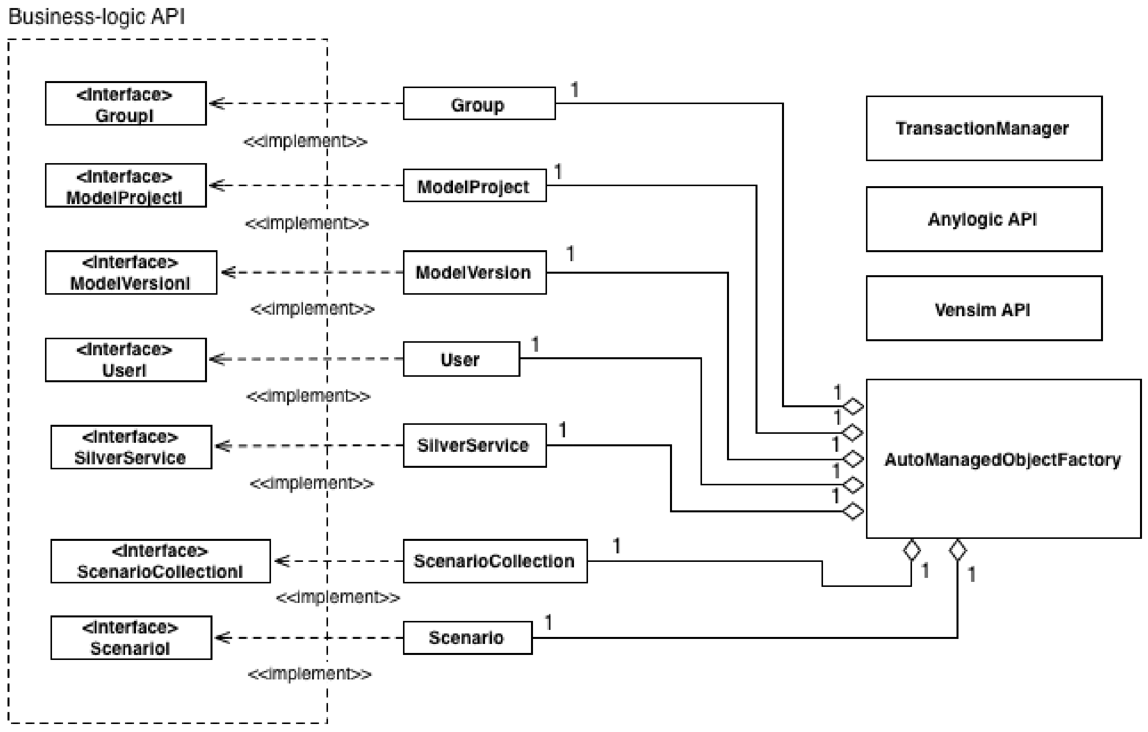


Figure 4.3: Business logic component design

The business logic component includes specific supports for two modeling software: Vensim and Anylogic. The details of their infrastructure are discussed in Section 4.4.1 and in Section 4.4.2.

4.2.4 Storage Component Design

This section discusses the storage component. The storage component is responsible for:

1. Managing the database based on configuration information provided from the business logic component
2. Storing all MPH information in a local database,
3. Storing all MPH information in a remote database,
4. Logging MPH modifications in the database, and
5. Forming XML representations for MPH structures for an MPH item or a complete log of changes.

Figure 4.4 displays the design of the storage component. At the top of the diagram, the StorageI interface gives the entire set of available operations at the storage level. The local database and remote database use exactly the same schema. The related tasks in storing MPH information in the storage layer are eventually accessed by clients through the business logic layer. Each data request from the client would be either require an one-step data access or a combination of results from multiple data access. To support the complexity of data operations in business logic layer, the StorageI interface is designed to provide flexibility to data access by providing fine-grained data operations that can either be executed independently or grouped as a batch operation. As a result, all operations defined in storage layer are to perform exactly one data operation about the data stored in the database. The supported data operations are 'CREATE', 'UPDATE' and 'DELETE'. This will allow the business logic component to combine two or more operations to fulfill a request from the client component.

The AbstractStorage class is a partial implementation of the StorageI interface that implements all data-related operations that are concerned with using SQL queries to perform data manipulation [13]. This also provides SQL queries to create or to destroy a database. All SQL queries used adhere to the SQL 2003 standard [18]. This standard is supported by Derby, the Java-based database project used by SILVER, as well as by other popular database projects [22]. This will allow the AbstractStorage class to be reused in building support from other database projects.

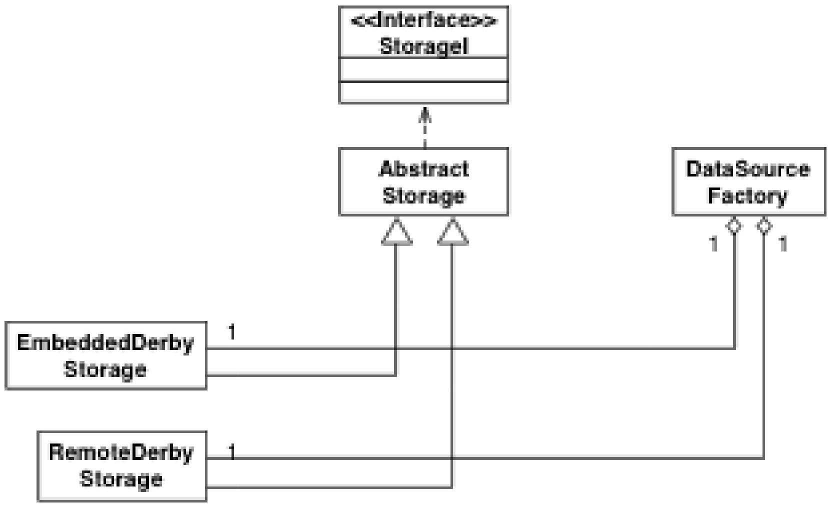


Figure 4.4: Storage component architecture

The AbstractStorage class is extended by two concrete Implementation classes called EmbeddedDerbyStorage and RemoteDerbyStorage. They are “driver classes” since they contain only information about database configuration that is specific to the context of connection and the type of database. For example, the EmbeddedDerbyStorage contains information about how to configure a Derby database on a hard drive in the user’s computer. The RemoteDerbyStorage contains information about how to configure a Derby database on a remote webserver through the Internet.

Finally, the “driver classes” are supported by the DataSourceFactory class that provides DataSource objects. The design of DataSourceFactory object follows the abstract factory pattern [26]. The goal of the abstract factory pattern is to provide ways to create families of related or dependent objects without specifying their concrete classes. In this case, the DataSourceFactory class provides various DataSource objects based on what has been requested by the “driver class”.

4.2.5 Database Design

As mentioned in Section 4.2.4, one of the responsibilities for the AbstractStorage class is creating a new database. In particular, this involves specifying tables and relationships to form the data schema in the database. An entity relationship diagram, or ER diagram, is shown in Appendix A; it displays the conceptual representation of data for SILVER.

In the entity relationship diagram, a table is created for each item in the MPH. This includes the SGROUP, MPROJECT, MVERSION, SCENARIOCOLLECTION and SCENARIO tables. Additional tables are added for the purpose of recording file attachments and change logging.

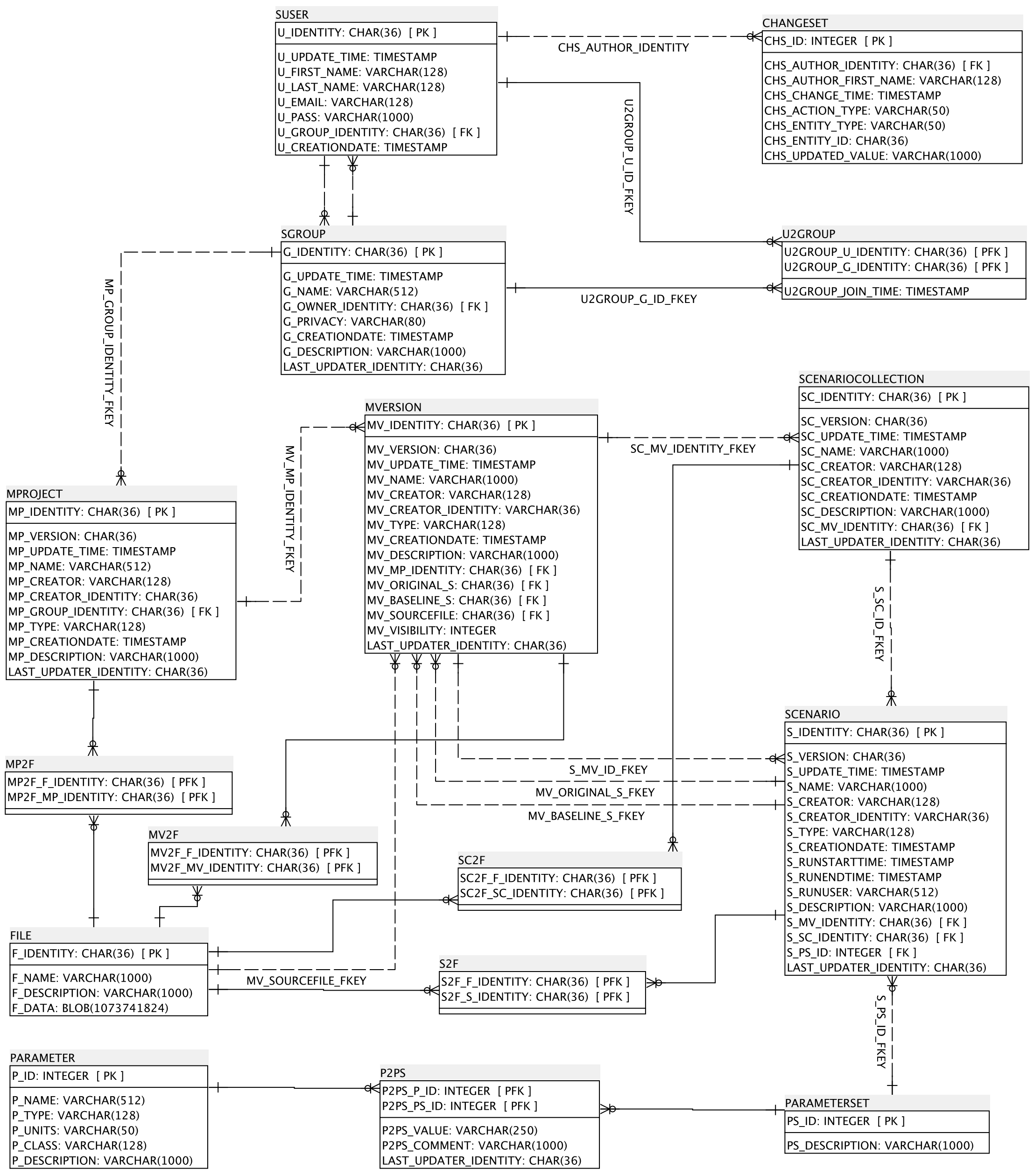

SUSER
U_IDENTITY: CHAR(36) [PK]
U_UPDATE_TIME: TIMESTAMP
U_FIRST_NAME: VARCHAR(128)
U_LAST_NAME: VARCHAR(128)
U_EMAIL: VARCHAR(128)
U_PASS: VARCHAR(1000)
U_GROUP_IDENTITY: CHAR(36) [FK]
U_CREATIONDATE: TIMESTAMP
CHS_AUTHOR_IDENTITY
CHANGESET
CHS_ID: INTEGER [PK]
CHS_AUTHOR_IDENTITY: CHAR(36) [FK]
CHS_AUTHOR_FIRST_NAME: VARCHAR(128)
CHS_CHANGE_TIME: TIMESTAMP
CHS_ACTION_TYPE: VARCHAR(50)
CHS_ENTITY_TYPE: VARCHAR(50)
CHS_ENTITY_ID: CHAR(36)
CHS_UPDATED_VALUE: VARCHAR(1000)
U2GROUP_U_ID_FKEY
SGROUP
G_IDENTITY: CHAR(36) [PK]
G_UPDATE_TIME: TIMESTAMP
G_NAME: VARCHAR(512)
G_OWNER_IDENTITY: CHAR(36) [FK]
G_PRIVACY: VARCHAR(80)
G_CREATIONDATE: TIMESTAMP
G_DESCRIPTION: VARCHAR(1000)
LAST_UPDATER_IDENTITY: CHAR(36)
U2GROUP
U2GROUP_U_IDENTITY: CHAR(36) [PFK]
U2GROUP_G_IDENTITY: CHAR(36) [PFK]
U2GROUP_JOIN_TIME: TIMESTAMP
U2GROUP_G_ID_FKEY
MP_GROUP_IDENTITY_FKEY
SCENARIOCOLLECTION
SC_IDENTITY: CHAR(36) [PK]
SC_VERSION: CHAR(36)
SC_UPDATE_TIME: TIMESTAMP
SC_NAME: VARCHAR(1000)
SC_CREATOR: VARCHAR(128)
SC_CREATOR_IDENTITY: VARCHAR(36)
SC_CREATIONDATE: TIMESTAMP
SC_DESCRIPTION: VARCHAR(1000)
SC_MV_IDENTITY: CHAR(36) [FK]
LAST_UPDATER_IDENTITY: CHAR(36)
MVERSION
MV_IDENTITY: CHAR(36) [PK]
MV_VERSION: CHAR(36)
MV_UPDATE_TIME: TIMESTAMP
MV_NAME: VARCHAR(1000)
MV_CREATOR: VARCHAR(128)
MV_CREATOR_IDENTITY: VARCHAR(36)
MV_TYPE: VARCHAR(128)
MV_CREATIONDATE: TIMESTAMP
MV_DESCRIPTION: VARCHAR(1000)
MV_MP_IDENTITY: CHAR(36) [FK]
MV_ORIGINAL_S: CHAR(36) [FK]
MV_BASELINE_S: CHAR(36) [FK]
MV_SOURCEFILE: CHAR(36) [FK]
MV_VISIBILITY: INTEGER
LAST_UPDATER_IDENTITY: CHAR(36)
SC_MV_IDENTITY_FKEY
MV_MP_IDENTITY_FKEY
MPROJECT
MP_IDENTITY: CHAR(36) [PK]
MP_VERSION: CHAR(36)
MP_UPDATE_TIME: TIMESTAMP
MP_NAME: VARCHAR(512)
MP_CREATOR: VARCHAR(128)
MP_CREATOR_IDENTITY: CHAR(36)
MP_GROUP_IDENTITY: CHAR(36) [FK]
MP_TYPE: VARCHAR(128)
MP_CREATIONDATE: TIMESTAMP
MP_DESCRIPTION: VARCHAR(1000)
LAST_UPDATER_IDENTITY: CHAR(36)
S_SC_ID_FKEY
SCENARIO
S_IDENTITY: CHAR(36) [PK]
S_VERSION: CHAR(36)
S_UPDATE_TIME: TIMESTAMP
S_NAME: VARCHAR(1000)
S_CREATOR: VARCHAR(128)
S_CREATOR_IDENTITY: VARCHAR(36)
S_TYPE: VARCHAR(128)
S_CREATIONDATE: TIMESTAMP
S_RUNSTARTTIME: TIMESTAMP
S_RUNENDTIME: TIMESTAMP
S_RUNUSER: VARCHAR(512)
S_DESCRIPTION: VARCHAR(1000)
S_MV_IDENTITY: CHAR(36) [FK]
S_SC_IDENTITY: CHAR(36) [FK]
S_PS_ID: INTEGER [FK]
LAST_UPDATER_IDENTITY: CHAR(36)
MP2F
MP2F_F_IDENTITY: CHAR(36) [PFK]
MP2F_MP_IDENTITY: CHAR(36) [PFK]
S_MV_ID_FKEY
MV_ORIGINAL_S_FKEY
MV_BASELINE_S_FKEY
MV2F
MV2F_F_IDENTITY: CHAR(36) [PFK]
MV2F_MV_IDENTITY: CHAR(36) [PFK]
SC2F
SC2F_F_IDENTITY: CHAR(36) [PFK]
SC2F_SC_IDENTITY: CHAR(36) [PFK]
FILE
F_IDENTITY: CHAR(36) [PK]
F_NAME: VARCHAR(1000)
F_DESCRIPTION: VARCHAR(1000)
F_DATA: BLOB(1073741824)
MV_SOURCEFILE_FKEY
S2F
S2F_F_IDENTITY: CHAR(36) [PFK]
S2F_S_IDENTITY: CHAR(36) [PFK]
S_PS_ID_FKEY
PARAMETER
P_ID: INTEGER [PK]
P_NAME: VARCHAR(512)
P_TYPE: VARCHAR(128)
P_UNITS: VARCHAR(50)
P_CLASS: VARCHAR(128)
P_DESCRIPTION: VARCHAR(1000)
P2PS
P2PS_P_ID: INTEGER [PFK]
P2PS_PS_ID: INTEGER [PFK]
P2PS_VALUE: VARCHAR(250)
P2PS_COMMENT: VARCHAR(1000)
LAST_UPDATER_IDENTITY: CHAR(36)
PARAMETERSET
PS_ID: INTEGER [PK]
PS_DESCRIPTION: VARCHAR(1000)

Figure 4.5: Database Entity Relationship

The design of the database and the responsibilities of each table are listed in Table 4.1. In Table 4.1, the primary key to each table is set to the type CHAR with size 36 in order to store an UUID. The primary key has been set to UUID for all tables except for the PARAMETER table. UUID assures that the same MPH data will be recognized and synchronized through different storage repositories.

4.2.6 Change Logging

Changes made in SILVER are recorded at the database level. The scope of changes includes creation, removal and modifications to each MPH item. The logging of changes is realized through database triggers. As part of the SQL specification, a database trigger can be appended or prepended to a single SQL query execution[18]. Therefore, the information about each modification related to MPH is recorded by a database trigger and stored to the CHANGESET table. The CHANGESET table contains the logging for an entire database.

4.2.7 Internationalization

All text-based information passed to the storage component is parsed by Base64 encoding [3]. The Base64 encoding allows encoding and decoding of any UTF8 based text to ANSI based text.

4.3 UI Component Design

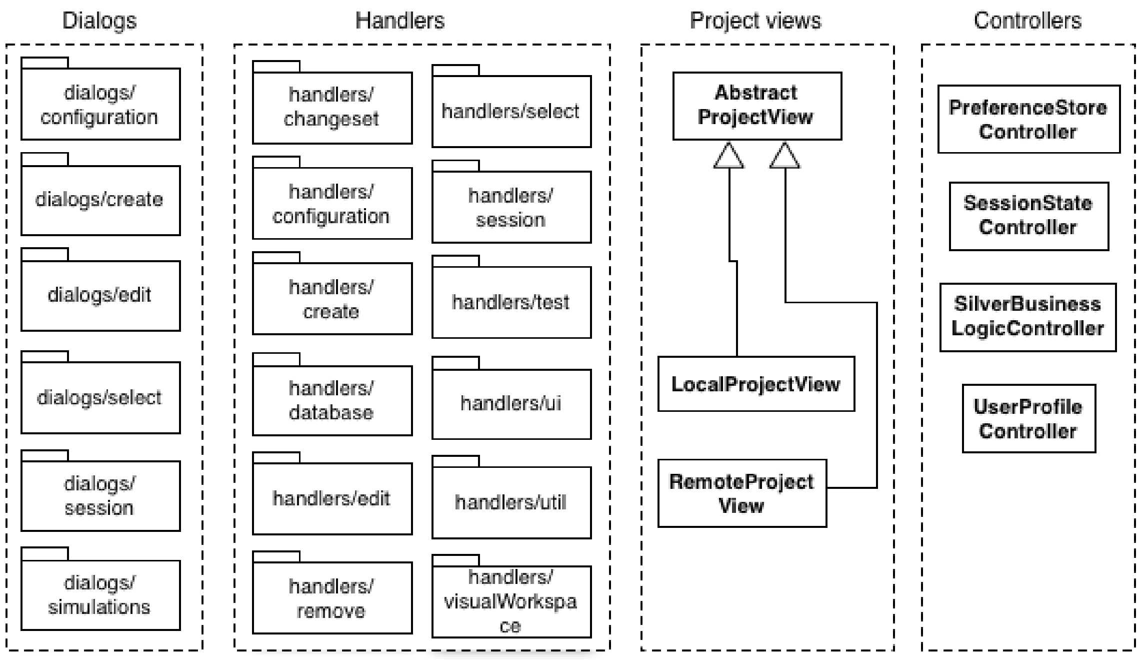


Figure 4.6: User Interface Component Design

The UI component is designed on top of the Eclipse rich client application framework [39]. The main classes and packages in the UI component are illustrated in Figure 4.6. At the left of the figure, the dialog classes support user actions in various contexts. The handler classes follow the command design pattern [26]. Each handler class responds to a command invocation and carries out the actual process. The command is an entity that can be placed at multiple locations in the UI; its corresponding handler is set in a script that is interpreted at system compile time. The project view classes are responsible for the two project views displayed at the left of the main user interface. The AbstractProjectView class contains a specification for the layout of the tree-structured interface. Each of the LocalProjectView class and RemoteProjectView class extends the AbstractProjectView class and initiate separate threads to allow concurrent access to various storage repositories.

4.3.1 Visual Workspace Component Design

As state in chapter 3, the visual workspace component is responsible to display change awareness. In fact, the visual workspace needs to re-construct the MPH display because the display space in the project view is too limited to display varied visual cue. The priorities of the goals for designing MPH re-construction in the visual workspace are:

- Provide overview of MPH
- Provide visual cues on MPH that convey change awareness information
- Allow user to interact with the visual workspace to explore change awareness information

A good way to provide an overview a visual structure is to provide its layout on a two dimentional surface. In this case, MPH can be displayed as tree structure. In order to fulfill the third requirement, the MPH can be envisioned using force-directed graph algorithm. In a force-directed graph, the position of all graph nodes is constantly updated until all edges reaches the balanced length. In SILVER, force-directed graph will allow users to adjust the position of all nodes in MPH to their best conveniency. In addition, since all nodes in a graph are displayed, it is easy to add visual cues directly to the graph to convey change awareness information.

As a result, the visual workspace component architecture consists of a graph-based visualization that represents parts of or the entire MPH. The component makes use of the Prefuse visualization toolkit to generate visualizations based on data input at application runtime [29]. The Prefuse toolkit follows the theoretical visualization framework [25].

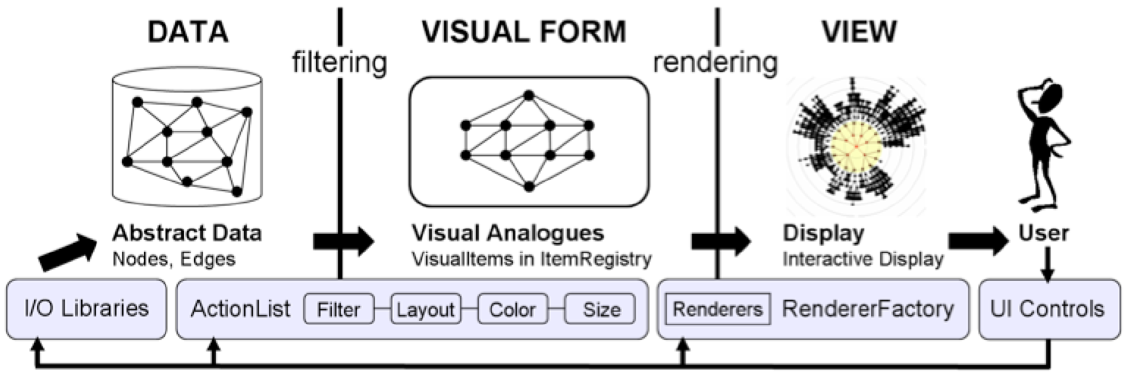


Figure 4.7: The Infovis Model in Prefuse, (figure from Heer 2005)

Figure 4.7 displays the visualization model provided by the Prefuse toolkit. The Infovis model decomposes design into a process consisting of several steps: abstract data representation, visual form and data mapping, and visual item display. The visual workspace integrates the process with the business logic infrastructure. The integrated model in the visual workspace is shown in Figure 4.8.

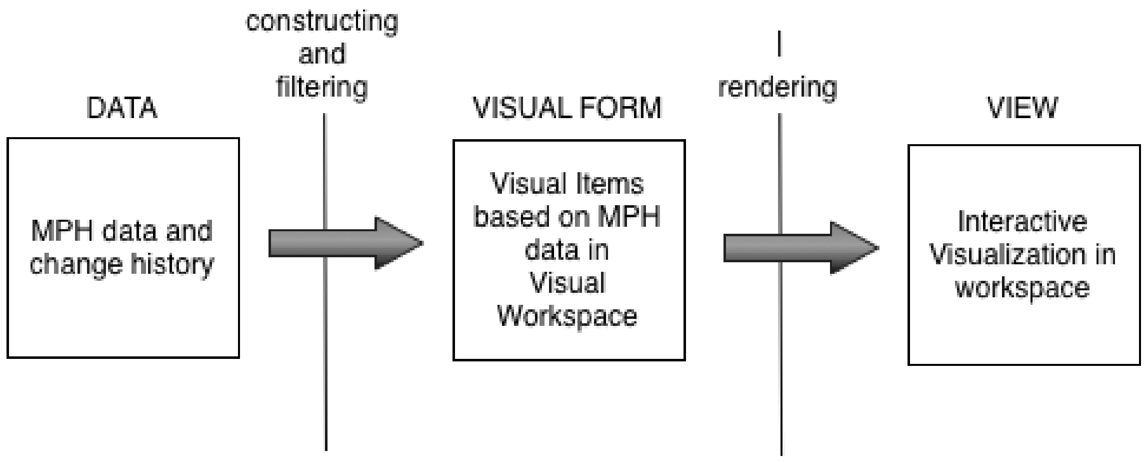

constructing
and
filtering
DATA
MPH data and change history
VISUAL FORM
Visual Items based on MPH data in Visual Workspace
rendering
VIEW
Interactive Visualization in workspace

Figure 4.8: Infovis Model in the Visual Workspace

To integrate the Prefuse visualization process in to SILVER, the visual workspace first re-creates the data from the business logic component in the in-memory database provided by Prefuse. For each record of the data, an object is created to help represent the abstract data in a visual environment. The visual workspace contains two sets of visualizations as part of the interface.

4.4 Simulation Software Support

This section talks about the architecture for the integration of external simulation software support in SILVER. In Section 4.2.3, the support has been described as a programming interface for Anylogic and Vensim models. Since both Anylogic and Vensim provide support in different ways, the corresponding supports are separated in different classes in SILVER.

The support for each software package consists of four sub-categories: model import, simulation execution, logging, and trajectory harvesting. Model import means importing all necessary information to create a baseline scenario that contains a parameter set, which is to be altered and analyzed for future use. Simulation execution requires information and files to run a simulation of a scenario based on the corresponding parameter values. Logging requires the states of the modeled system to be recorded along with the simulation. Normally a state is represented by certain datasets within the model system. The dataset is important feedback as the assumptions made prior to running the simulation, and for the consequences of those assumptions. It is important to locate the dataset and allow users to share it along with further documentations.

Each of the following sections will break the discussion into four steps with respect to the sub-categories above.

4.4.1 Vensim Support

Vensim support consists of parameter logging done through the VENDLL runtime library. VENDLL runtime library provides API that provides programming access to Vensim functionalities without the support of the Vensim modeling software.

1. *Model import.* The model import requires a Vensim published model (vpm file) as the model source file. All types of variables in the model can be accessed through the Vensim runtime library.

2. *Simulation running.* Executing a Vensim simulation requires the user to enter a name

for the simulation run.

3. *Logging.* Datasets are automatically generated during the Vensim simulation. The Vensim simulation exports datasets to a file by the end of the simulation. The exported simulation is called vdf file.

4. *Trajectory harvesting.* A vdf file that contains the dataset is generated after the Vensim simulation is executed.

With support of the Vensim programming interface and Vensim runtime library, most support procedures can be accomplished using Vensim API.

4.4.2 AnyLogic Support

The Anylogic support includes taking a screen shot of the simulation result and logging data sets to exported Anylogic simulation. An exported Anylogic simulation is an self-packed Java Applet that executes a simulation without the support of Anylogic software. Both screenshot and logging is realized through AspectJ load-time weaving to the simulation applet. AspectJ is an aspect-oriented extension to the Java programming language. [34]. It allows SILVER to control an exported simulation based on the user's needs.

1. *Model import.* The model support for Anylogic takes the model source file (in the form of an XML-based .alp file) as well as the corresponding exported simulation.

2. *Simulation.* The simulation will execute the exported simulation and override parameters values with what has been set in the scenario prior to simulation start.

3. *Logging.* Logging is realized through keeping track of DataSet objects in Anylogic. DataSet objects are used to keep samples of parameter values over each time frame.

4. *Trajectory harvesting.* The DataSet values will be saved to a csv file after the simulation is finished. A csv file contains comma-separated values in plain-text form.

Figure 4.9 gives a concise overview of how an Anylogic simulation is supported in SILVER. Each time a simulation is initiated, SILVER exports the simulation applet and an XML file that consists of parameter values to the hard drive. The exported simulation applet is then controlled by AspectJ load-time weaving to perform necessary changes and simulation.

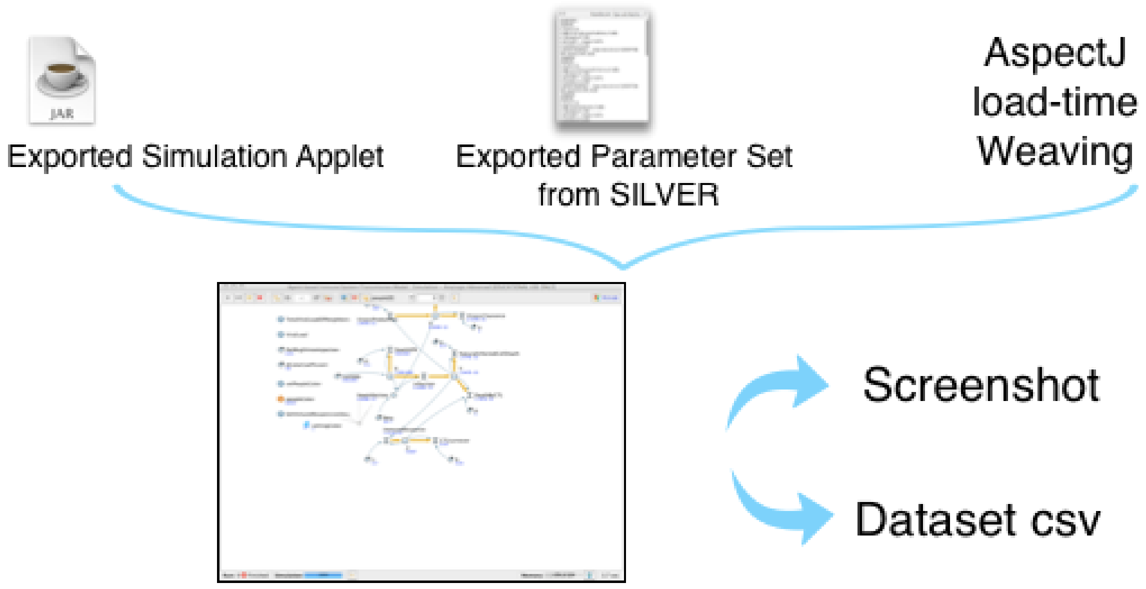

Exported Simulation Applet
Exported Parameter Set from SILVER
AspectJ load-time Weaving
Screenshot
Dataset csv

Figure 4.9: Anylogic Support in SILVER

Table Name	Specification and Responsibilities
SUSER	• all information for other user to recognize a person. • required information to specify identity of a user in the collaborative modeling process.
SGROUP	• represent a work group • can specify the creator of the group
U2GROUP	• specifies how many users are participating in the group
CHANGESET	• record the time, the type and the action detail of the change • specify the associated person that has made the change
MPROJECT	• the group in which this project is residing • original/baseline model version • associated files
MVERSION	• the model project for which this is a version • baseline scenario • associated files
SCENARIOCOLLECTION	• the model version with which this scenario collection is associated • associated files
SCENARIO	• the belonging scenario collection with which this scenario is associated • the model source file and related files required to run the simulation • the associated parameter set
PARAMETERSET	• the scenario it belongs to
P2PS	• specify the parameter set in the PARAMETERSET table and its associated parameter values in the PARAMETER table
PARAMETER	• a parameter value and unit
FILE	• file name • file content
MP2F	• specify a model project and its associated file
MV2F	• specify a model version and its associated file
SC2F	• specify a scenario collection and its associated file
S2F	• specify a scenario and its associated file

Table 4.1: Database Table Specification

CHAPTER 5

THE VISUAL WORKSPACE

This chapter introduces the collaborative workspace that has been developed as part of SILVER. The collaborative wolrkspace has been built using the GUI component infrastructure discussed in Chapter 4, and has been designed to apply part of the change-awareness framework to the modeling process hierarchy detailed in Chapter 3.

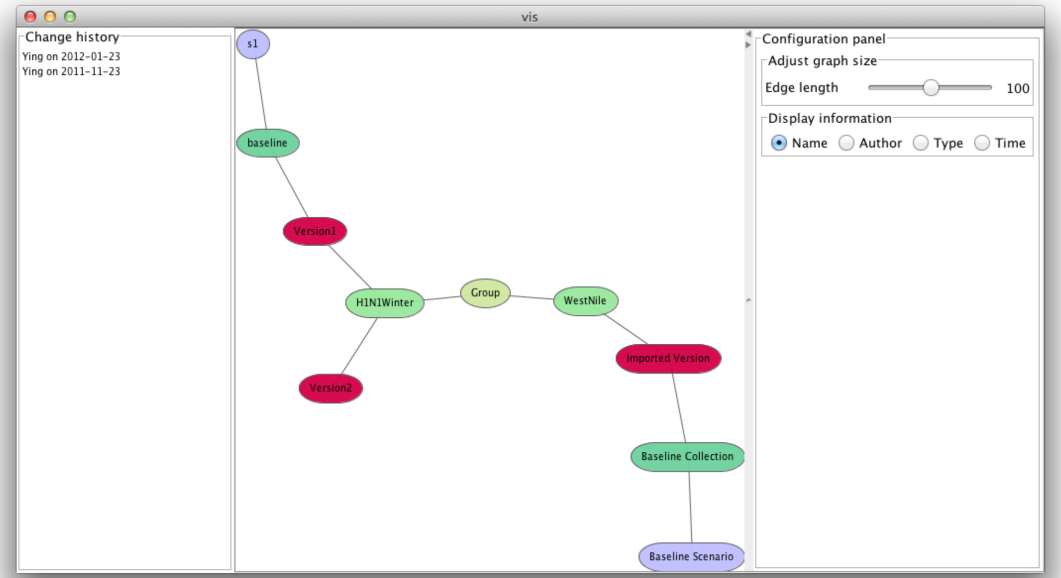
The collaborative workspace is divided into two components. Section 5.1 discusses a visual workspace that contains iterative visualizations to support part of change awareness; Section 5.2 discusses a ChangeSet editor that displays a history of changes.

5.1 The Visual Workspace

The visual workspace combines information related to user activity with the visual representation of the MPH. The goal of building the visual workspace is to reduce the effort required in activites such as exploring and collecting change awareness information from an established or a work-in-progress model in SILVER. This involves designing a visual representation of the MPH, additional visual embodiments, and control widgets.

5.1.1 Visualizing the Model Hierarchy

A visual workspace is generated at runtime once a user selects an item and chooses 'show visual workspace' in a model repository view. The visual workspace is generated at runtime based on the user's selection. As shown in Figure 5.1, the middle of the visual workspace depicts a MPH structure containing a group and two model projects in a graph based visualization, where nodes represent items of the MPH and edges represent associations between those items. The nodes can be dragged around, and open detailed information about the item upon mouse double-click. In this visualization, network members are presented using both their names and a color. The color indicates the type of item.



Ying on 2012-01-23
Ying on 2011-11-23

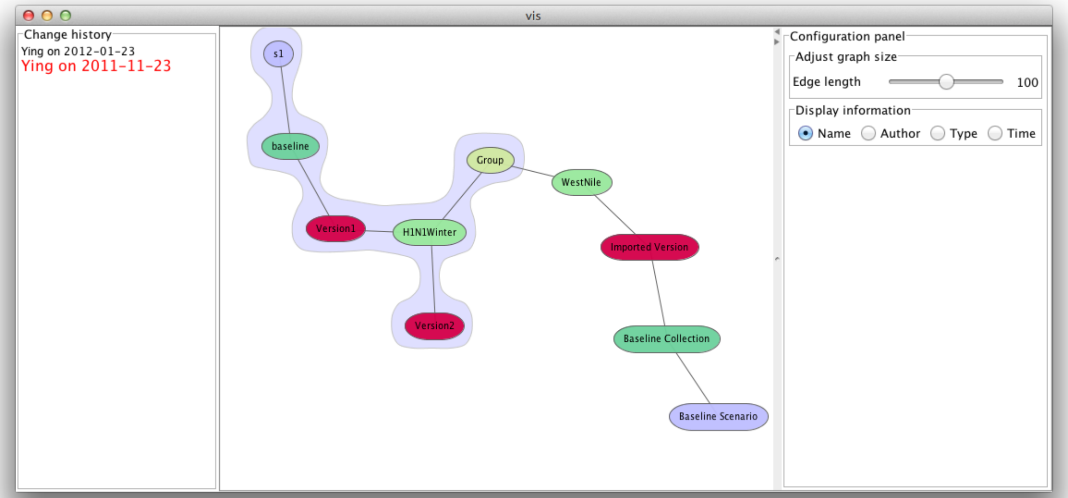
Edge length 100

(● Name) ○ Author ○ Type ○ Time

Figure 5.1: The Visual Workspace

The graph based network is generated by the Prefuse toolkit and uses force-directed layout to maintain the relative display location for each node [29, 33]. The network layout is computed using a spring-embedding (force-directed) algorithm, in which the edges act as springs and both draw distant neighboring towards itself but repels nearby neighbouring nodes [33]. The network layout of the representation is computed in real time. When a node has been moved by a modeler, the displacement of the node will exert a force on its neighbouring nodes. The edges in the network will react to the deflection and trigger (dissipative) readjustment until the system re-establishes a static equilibrium. This approach grants users the freedom to move the structure into a convenient, memorable, and aesthetically pleasing arrangement.

The length of graph edge in the visual workspace can be adjusted by the control widgets at the right side of the visual workspace. This can be used to enlarge or shrink the overall graph based on the display area available. On top of the edge length adjustment widget, four radio buttons are provided to allow users to choose one of “name”, “type”, “author” and “time”. After a user makes a selection, each graph node displays the new information within itself.



Change history
Ying on 2012-01-23
Ying on 2011-11-23

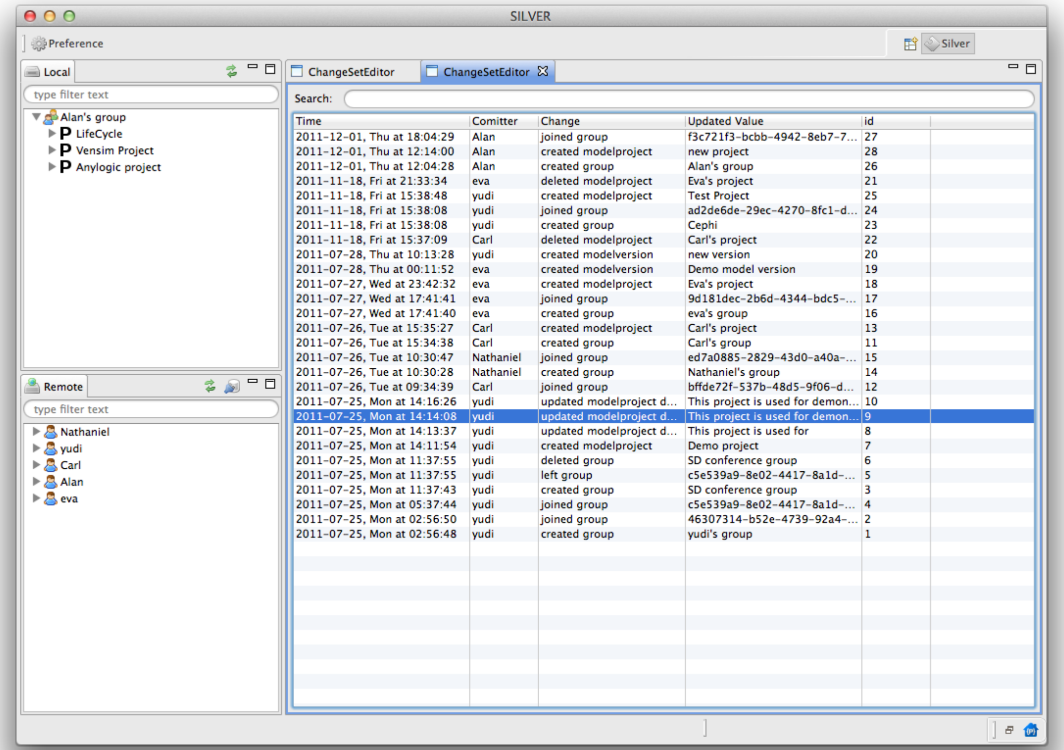
Configuration panel
Adjust graph size
Edge length 100
Display information
[✓ Name] Author Type Time

Figure 5.2: Highlighting Changes

5.1.2 Displaying ChangeSet

At the left of the graph visualization of the MPH is a timeline view represented via fish-eye visualization. The timeline lists dates that contains changes in a column and arranges them in decending in chronological order. With the fish-eye visualization, the dates that are adjacent to the mouse pointer on the screen are magnified based on their proximity to the mouse position. The closest date to the mouse pointer will be the bigger than the remainder of the dates and therefore would be more readily selected with a left mouse-click.

A left mouse-click on a highlighted date highlights the items in the MPH that have been changed on that day. In Figure 5.2, a few nodes were highlighted in the graph visualization of the MPH to indicate that they have been changed. The grouping technique used on top of the visualization is called Bubble Sets [16]. According to Collins, Bubble Sets is a visualization technique for data that has both a primary data relation with a semantically significant spatial organization and a significant set membership relation in which members of the same set are not necessarily adjacent in the primary layout. In the context of a dynamic environment, the contour of Bubble Sets is constantly updated to avoid overlapping and blocking nodes that not members to the set. For example, in Figure 5.2, the purple set groups five nodes that were changed by the specific date while avoiding



Time	Comitter	Change	Updated Value	id
2011-12-01, Thu at 18:04:29	Alan	joined group	f3c721f3-bcbb-4942-8eb7-7...	27
2011-12-01, Thu at 12:14:00	Alan	created modelproject	new project	28
2011-12-01, Thu at 12:04:28	Alan	created group	Alan's group	26
2011-11-18, Fri at 21:33:34	eva	deleted modelproject	Eva's project	21
2011-11-18, Fri at 15:38:48	yudi	created modelproject	Test Project	25
2011-11-18, Fri at 15:38:08	yudi	joined group	ad2de6de-29ec-4270-8fc1-d...	24
2011-11-18, Fri at 15:38:08	yudi	created group	Cephi	23
2011-11-18, Fri at 15:37:09	Carl	deleted modelproject	Carl's project	22
2011-07-28, Thu at 10:13:28	yudi	created modelversion	new version	20
2011-07-28, Thu at 00:11:52	eva	created modelversion	Demo model version	19
2011-07-27, Wed at 23:42:32	eva	created modelproject	Eva's project	18
2011-07-27, Wed at 17:41:41	eva	joined group	9d181dec-2b6d-4344-bdc5-...	17
2011-07-27, Wed at 17:41:40	eva	created group	eva's group	16
2011-07-26, Tue at 15:35:27	Carl	created modelproject	Carl's project	13
2011-07-26, Tue at 15:34:38	Carl	created group	Carl's group	11
2011-07-26, Tue at 10:30:47	Nathaniel	joined group	ed7a0885-2829-43d0-a40a-...	15
2011-07-26, Tue at 10:30:28	Nathaniel	created group	Nathaniel's group	14
2011-07-26, Tue at 09:34:39	Carl	joined group	bffde72f-537b-48d5-9f06-d...	12
2011-07-25, Mon at 14:16:26	yudi	updated modelproject d...	This project is used for demon...	10
2011-07-25, Mon at 14:14:08	yudi	updated modelproject d...	This project is used for demon...	9
2011-07-25, Mon at 14:13:37	yudi	updated modelproject d...	This project is used for	8
2011-07-25, Mon at 14:11:54	yudi	created modelproject	Demo project	7
2011-07-25, Mon at 11:37:55	yudi	deleted group	SD conference group	6
2011-07-25, Mon at 11:37:55	yudi	left group	c5e539a9-8e02-4417-8a1d-...	5
2011-07-25, Mon at 11:37:43	yudi	created group	SD conference group	3
2011-07-25, Mon at 05:37:44	yudi	joined group	c5e539a9-8e02-4417-8a1d-...	4
2011-07-25, Mon at 02:56:50	yudi	joined group	46307314-b52e-4739-92a4-...	2
2011-07-25, Mon at 02:56:48	yudi	created group	yudi's group	1

Figure 5.3: ChangeSet editor

the node that would otherwise be likely to be included among the highlighted nodes.

5.2 ChangeSet Editor

As shown in Figure 5.3, the ChangeSet editor contains a list of changes associated with an entire modeling process hierachy in a model repository. The ChangeSet editor can be activated by 'right-mouse clicking' in the project view and select 'Show Change Sets' from the pop-up menu. The editor then appears in the editor area and shows five categories of metadata associated with each change within the storage. They are: "Time", "Committer", "Change", "Updated Value" and "id". Each category is labeled by particular column in the editor. The user may choose to re-arrange the editor display by selecting a particular column. The ChangeSet editor also supports user in selecting and opening particular entities in the modeling process hierarchy that were involved in the change.

CHAPTER 6

THE MODELING PROCESS WITH SILVER

This chapter provides a detailed guide on the use of SILVER to assist individual and collaborative modeling process. The guide assumes that the reader is familiar with modeling process via one of the two modeling tools, so it provides separate discussions for modeling with Anylogic and with Vensim. Additionally, the guide assumes that the reader has some familiarity with domain terminologies defined by model repository concepts outlined in chapter 2, such as model project and the use of the model version and scenario.

6.1 Working with an Anylogic model

This section will provide instructions on the basics of performing individual and collaborative modeling with SILVER and Anylogic. Although the steps in creating Agent-based model and in System Dynamics model may be different within Anylogic, the created models do not require extra steps to be loaded to SILVER.

Steps for creating a simple Anylogic model and importing it into SILVER:

(a) Creating a new Anylogic model

Start up Anylogic, and create a model in Anylogic, as shown in Figure 6.1.

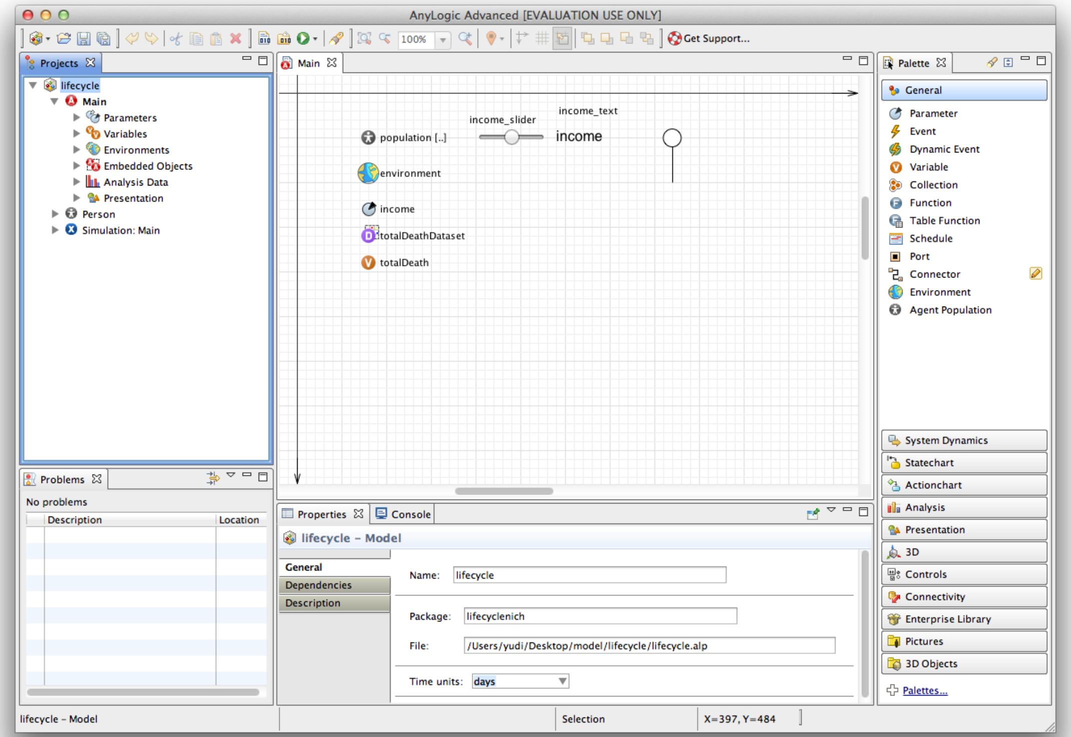
(b) Export the Anylogic simulation

Use the export functionalities in Anylogic, choose “export to Java Applet”.

(c) Load the existing model and simulation by creating a new Model Project or a Model Version on SILVER. Because these processes differ, we describe each separately.

(c-1) Create a model project

In SILVER, create a new model project by right-clicking in the project view and choose “add model project” in the context menu. In the “Add Model Project” dialog, the user can specify the path to the .alp file for the model source field and the path to the exported applet for the *exported applet field*. The user is required to give a name and a description for the



Name: lifecycle
Package: lifecyclenich
File: /Users/yudi/Desktop/model/lifecycle/lifecycle.alp
Time units: days

Figure 6.1: Create an Anylogic model

model project. It is optional to specify additional files on the hard drive that are related to the project. After user has filled in the necessary information (shown in Figure 6.2) and clicked "OK" button in the "Add Model Project" dialog, a model project, its associated baseline model version, and a baseline scenario will be created.

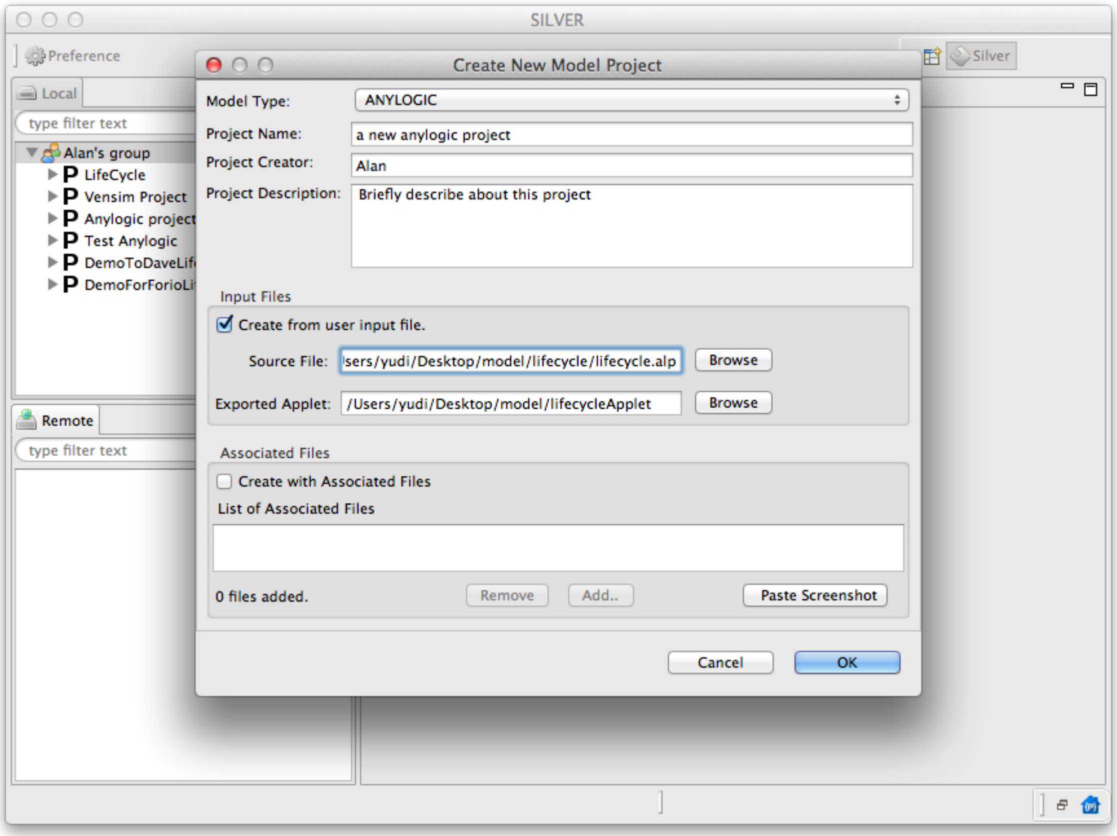
(c-2) Browse to baseline scenario

The user can expand the model project and reveal the associated model versions by double-clicking on the project. From the baseline model version that has been revealed, double-click on it again will reveal the associated baseline scenario.

(d) Run simulation in the scenario panel

Once the user selects the baseline scenario and double-clicks on it in the project view, detailed information about the scenario is shown in the information panel to the right of the project view. The "run simulation" button at the bottom of the panel will execute the Anylogic simulation. The simulation will be executed in a different system process.

(e) Loading simulation trajectories



Model Type: ANYLOGIC
Project Name: a new anylogic project
Project Creator: Alan
Project Description: Briefly describe about this project

Input Files
☑ Create from user input file.
Source File: !sers/yudi/Desktop/model/lifecycle/lifecycle.alp [Browse]
Exported Applet: /Users/yudi/Desktop/model/lifecycleApplet [Browse]

Associated Files
☐ Create with Associated Files
List of Associated Files

0 files added. [Remove] [Add..] [Paste Screenshot]

[Cancel] [OK]

Figure 6.2: Import an Anylogic model to SILVER

Once the Anylogic simulation is finished, a screenshot of the simulation and the values of DataSet objects are SILVER will be saved in separate files and loaded as associated files to the scenario where the simulation was executed. The user may double-click on the files to open them with the software determined by the operating system.

6.2 Working with a Vensim Model

This section provides instructions on the basics for getting started with individual and collaborative modeling with SILVER and Vensim.

Steps in creating a simple Vensim model and importing it into SILVER.

(a) In Vensim, create a new Vensim model

Start Vensim and build a vensim model with the tools provided.

(b) Setup the Vensim dynamic-link library with SILVER

The Vensim dynamic-link library is loaded through preferences page that can be accessed through “preferences” button on the toolbar.

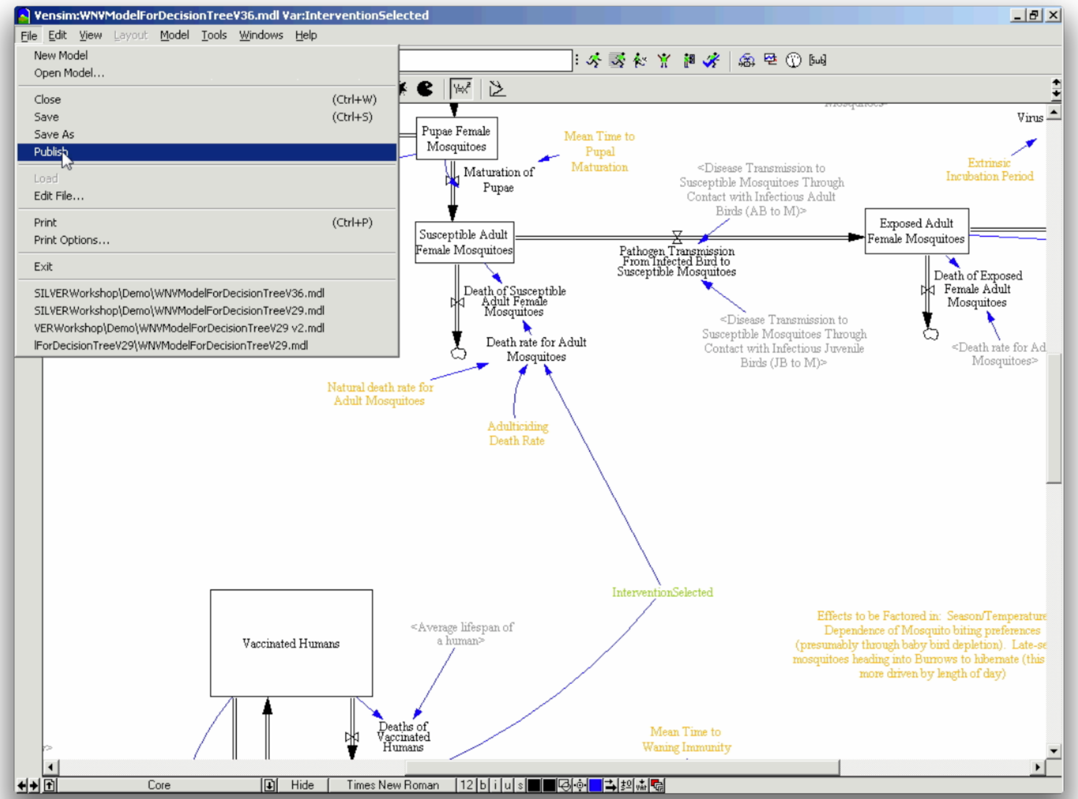

Vensim:WNVModelForDecisionTreeV36.mdl Var:InterventionSelected
File Edit View Layout Model Tools Windows Help
New Model
Open Model...
Close (Ctrl+W)
Save (Ctrl+S)
Save As
Publish
Load
Edit File...
Print (Ctrl+P)
Print Options...
Exit
SILVERWorkshop\Demo\WNVModelForDecisionTreeV36.mdl
SILVERWorkshop\Demo\WNVModelForDecisionTreeV29.mdl
VERWorkshop\Demo\WNVModelForDecisionTreeV29 v2.mdl
lForDecisionTreeV29\WNVModelForDecisionTreeV29.mdl
Pupae Female Mosquitoes
Mean Time to Pupal Maturation
Maturation of Pupae
<Disease Transmission to Susceptible Mosquitoes Through Contact with Infectious Adult Birds (AB to M)>
Virus
Extrinsic Incubation Period
Susceptible Adult Female Mosquitoes
Pathogen Transmission From Infected Bird to Susceptible Mosquitoes
Exposed Adult Female Mosquitoes
Death of Susceptible Adult Female Mosquitoes
Death of Exposed Female Adult Mosquitoes
<Disease Transmission to Susceptible Mosquitoes Through Contact with Infectious Juvenile Birds (JB to M)>
Death rate for Adult Mosquitoes
<Death rate for Ad Mosquitoes>
Natural death rate for Adult Mosquitoes
Adulticiding Death Rate
InterventionSelected
Vaccinated Humans
<Average lifespan of a human>
Effects to be Factored in: Season/Temperature Dependence of Mosquito biting preferences (presumably through baby bird depletion). Late-se mosquitoes heading into Burrows to hibernate (this more driven by length of day)
Deaths of Vaccinated Humans
Mean Time to Waning Immunity
Core
Hide
Times New Roman
12

Figure 6.3: Publish an Vensim model

(c) In Vensim, with the model finished, choose 'Publish' from the File menu, as shown in Figure 6.3. Within the dialog, determin the file name and save the published model to hard drive.

(d) Loading an existing model by creating a new Model Project or a new Model Version.

As has been explained in Section 6.1, create a new model project by right-clicking in the project view and choose "Add Model Project" in the context menu. In the "Add Model Project" dialog, the information required to be filled by a user is similar to that for adding a model project by importing it into an Anylogic model. Instead, the user can specify the path to the .vpm file for the model source and leave the path to the exported applet as blank.

(e) Run simulation in the scenario panel

Once the user runs the Vensim model through the scenario panel, the Vensim runtime will load the Vensim model, load parameter set values from the scenario to the simulation, and ask user to assign a name to the simulation run.

(f) Loading simulation trajectories

Vensim dynamic-link library will generate .vdf files that allows Vensim users to share simulation results with other collaborators. The .vdf file generated will be loaded and saved in SILVER under the scenario that has been opened.

6.3 Working with Models Stored Remotely

This section provides instructions on the basics for working with models that are stored on the remote model repository in SILVER.

The following steps describes sharing a model on remote model repository; p

(a) Uploading an existing model to remote model repository by creating a new Model Project and a new Model Version.

As has been explained in Section 6.1 and in Section 6.2, in the remote project view, create a new model project by right-clicking in the project view and choose "Add Model Project" in the context menu. User can decide whether to add Anylogic model project, Vensim model project, or simply a generic model project that does not have a model source.

(b) Run a remotely added simulation in the scenario panel

Once the model project is created, all users that are connected to the remote repository are able to run the model. The criteria for a user to run a model are the same compared to running the model created in local model repository. For example, If the model is created in Vensim, the user who tries to run the model need to have Vensim DLL library configured in SILVER before running the simulation.

(c) Sharing and loading simulation trajectories

Once a simulation is executed and stopped, SILVER will harvest the trajectory files (.vdf for Vensim models, .csv for Anylogic models) and upload them to the scenario where the simulation has been triggered. Once the trajectory files have been uploaded, all users that are connected to the remote model repository may simply re-open the scenario information panel and double-click on the file the user wished to see.

6.3.1 Adding External Documents

This section provides instructions for importing or sharing documents with respect to MPH entities during the modeling collaboration.

Steps in adding an external file to an MPH entity.

(a) Create or select the entity in the modeling process hierarchy

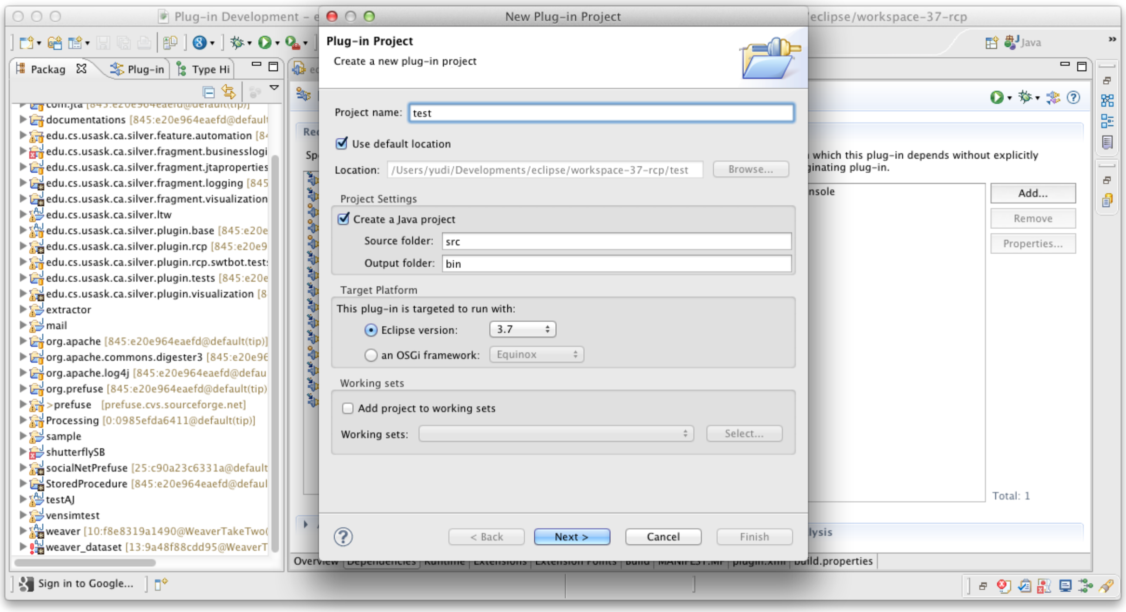
(b) Double-click on the entity to open the information panel about the entity to the right of MPH view.

(c) In the “associated files” section of the information panel, click “Add” to add one or more associated files to the entity.

CHAPTER 7

EVALUATION

SILVER has been informally evaluated based on a number of criteria that are important to modelers and collaborators. With respect to modelers, the criteria include effort reduction, modeling support, flexibility and extensibility. From the user‘s perspective, some analysis should be done to determine the adequacy of the awareness information provided by the ChangeSets and the visualization of that awareness information. The remainder of this chapter will discuss the evaluation process in general before delving into each of the evaluation criteria in detail.


New Plug-in Project

Plug-in Project

Create a new plug-in project

Project name: test

[✓] Use default location

Location: /Users/yudi/Developments/eclipse/workspace-37-rcp/test

Project Settings

[✓] Create a Java project

Source folder: src

Output folder: bin

Target Platform

This plug-in is targeted to run with:

[●] Eclipse version: 3.7

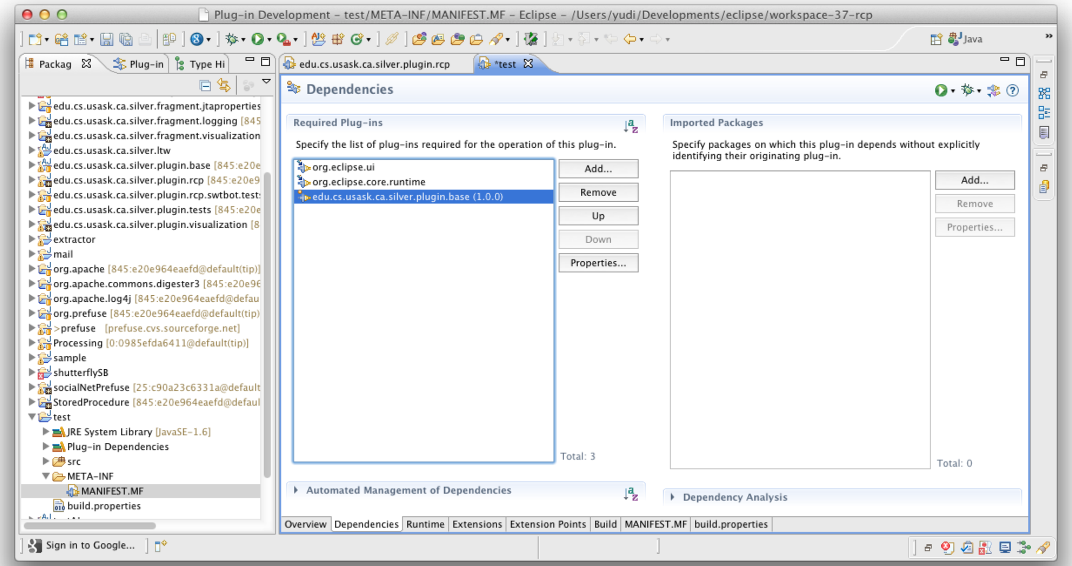
[○] an OSGi framework: Equinox

Working sets

[ ] Add project to working sets

Working sets:

Figure 7.1: Create test plugin


Plug-in Development – test/META-INF/MANIFEST.MF – Eclipse – /Users/yudi/Developments/eclipse/workspace-37-rcp

edu.cs.usask.ca.silver.fragment.jtaproperties
edu.cs.usask.ca.silver.fragment.logging [845
edu.cs.usask.ca.silver.fragment.visualization
edu.cs.usask.ca.silver.ltw
edu.cs.usask.ca.silver.plugin.base [845:e20e
edu.cs.usask.ca.silver.plugin.rcp [845:e20e9
edu.cs.usask.ca.silver.plugin.rcp.swtbot.test:
edu.cs.usask.ca.silver.plugin.tests [845:e20e
edu.cs.usask.ca.silver.plugin.visualization [8
extractor
mail
org.apache [845:e20e964eaefd@default(tip)]
org.apache.commons.digester3 [845:e20e96
org.apache.log4j [845:e20e964eaefd@defau
org.prefuse [845:e20e964eaefd@default(tip)
>prefuse [prefuse.cvs.sourceforge.net]
Processing [0:0985efda6411@default(tip)]
sample
shutterflySB
socialNetPrefuse [25:c90a23c6331a@default
StoredProcedure [845:e20e964eaefd@defaul
test
JRE System Library [JavaSE-1.6]
Plug-in Dependencies
src
META-INF
MANIFEST.MF
build.properties

Dependencies

Required Plug-ins
Specify the list of plug-ins required for the operation of this plug-in.
org.eclipse.ui
org.eclipse.core.runtime
edu.cs.usask.ca.silver.plugin.base (1.0.0)
Total: 3

Imported Packages
Specify packages on which this plug-in depends without explicitly identifying their originating plug-in.
Total: 0

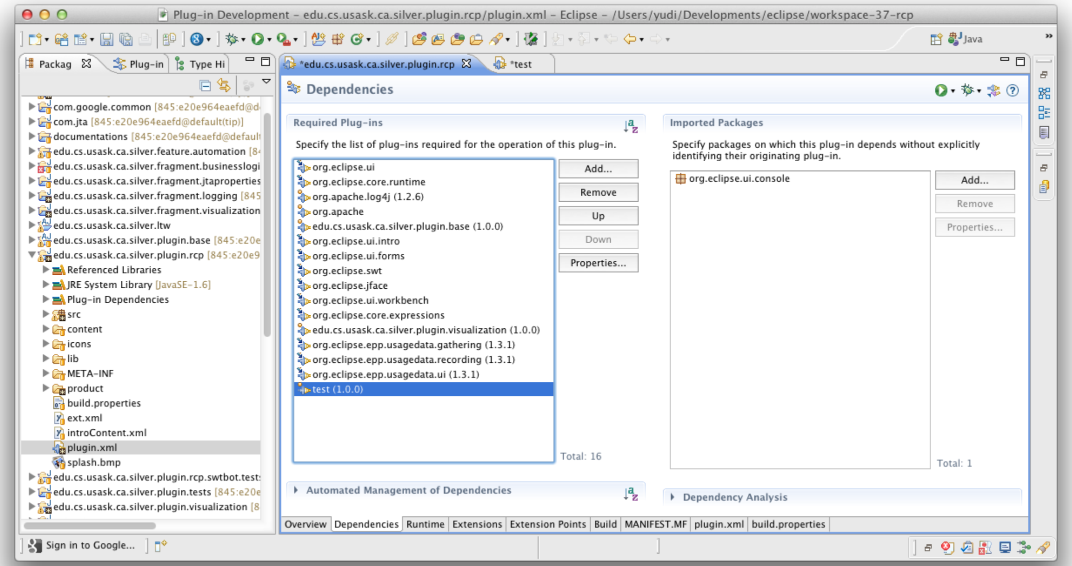
Figure 7.2: Import base plugin in test plugin

7.1 Evaluation Process

To assess SILVER, two different evaluations[1] were performed. The first evaluation involved a user participating in a distributed modeling process to determine how easy it is to participate and collaborate in a modeling process using SILVER and other modeling software. More specifically, the evaluator was asked to perform constructive tasks under a few pre-built modeling projects. In building the model versions or scenarios based on what others have contributed, the performance and limitations of modeling with SILVER could also be determined.

The second evaluation involved looking for information related to changes within two different model projects that had already been formulated by a large team. The visual workspace and changeset history were both used in contextual tasks in modeling. The evaluation provided information about difficulty reduction when using SILVER, for collaborative modeling, when compared to conducting the same process without SILVER.

[1]Behavioural ethics approval number: BSC 2001-192



Plug-in Development – edu.cs.usask.ca.silver.plugin.rcp/plugin.xml – Eclipse – /Users/yudi/Developments/eclipse/workspace-37-rcp

```
▶ com.google.common [845:e20e964eaefd@d
▶ com.jta [845:e20e964eaefd@default(tip)]
▶ documentations [845:e20e964eaefd@default
▶ edu.cs.usask.ca.silver.feature.automation [8
▶ edu.cs.usask.ca.silver.fragment.businesslogi
▶ edu.cs.usask.ca.silver.fragment.jtaproperties
▶ edu.cs.usask.ca.silver.fragment.logging [845
▶ edu.cs.usask.ca.silver.fragment.visualization
▶ edu.cs.usask.ca.silver.ltw
▶ edu.cs.usask.ca.silver.plugin.base [845:e20e
▼ edu.cs.usask.ca.silver.plugin.rcp [845:e20e9
    ▶ Referenced Libraries
    ▶ JRE System Library [JavaSE-1.6]
    ▶ Plug-in Dependencies
    ▶ src
    ▶ content
    ▶ icons
    ▶ lib
    ▶ META-INF
    ▶ product
      build.properties
      ext.xml
      introContent.xml
      plugin.xml
      splash.bmp
▶ edu.cs.usask.ca.silver.plugin.rcp.swtbot.test:
▶ edu.cs.usask.ca.silver.plugin.tests [845:e20e
▶ edu.cs.usask.ca.silver.plugin.visualization [8
```

Dependencies

Required Plug-ins

Specify the list of plug-ins required for the operation of this plug-in.

```
org.eclipse.ui
org.eclipse.core.runtime
org.apache.log4j (1.2.6)
org.apache
edu.cs.usask.ca.silver.plugin.base (1.0.0)
org.eclipse.ui.intro
org.eclipse.ui.forms
org.eclipse.swt
org.eclipse.jface
org.eclipse.ui.workbench
org.eclipse.core.expressions
edu.cs.usask.ca.silver.plugin.visualization (1.0.0)
org.eclipse.epp.usagedata.gathering (1.3.1)
org.eclipse.epp.usagedata.recording (1.3.1)
org.eclipse.epp.usagedata.ui (1.3.1)
test (1.0.0)
```

Total: 16

Imported Packages

Specify packages on which this plug-in depends without explicitly identifying their originating plug-in.

```
org.eclipse.ui.console
```

Total: 1

Figure 7.3: Import test plugin in SILVER

7.2 Extensibility

Extensibility refers to the ability of the SILVER architecture and component set to be extended to add new features that are not provided in its standard form. This includes adding supports for additional modeling software, extending ad hoc package-specific features, and changing the communication infrastructure. Support to additional modeling software can be provided by implementing modeling process hierarchy APIs. Depending on the nature of the modeling software and underlying modeling techniques applied, this may require different levels of complexity. As mentioned previously, the business logic component can be extended to help specify particular needs in each stage of modeling process hierarchy. To ease the development, the business logic component provides separate data structures for the modeling process hierarchy and change history. The data structures can be used in the creation of new interactions and visualizations to support awareness in collaboration. Finally, because SILVER is a component based system that is built on top of Eclipse OSGi runtime, additional plugins that address missing functionalities can be added easily to the system. To illustrate how to add a plugin to SILVER, an example to add an empty plugin is provided.

7.2.1 Example to Extend SILVER

This section provides a simple guide to show how to extend SILVER by adding an empty plugin that has access to business logic API.

- A Download the source code from SILVER project website to hard drive [9].
- Eclipse RCP IDE, load up the source code by importing projects to workspace [4].
- create a 'Plugin-in project' and name it 'test', as shown in Figure 7.1; select 'No' when encountered question 'Would you like to create a rich client application?'; click 'Finish'.
- Once added, the 'MENIFEST.MF' file under the 'test' project is automatically opened; Open 'Dependencies' tab, click 'Add...' to trigger a dialogue that allows us to add an existing plugin to the 'required plugins', as shown in Figure 7.2. Adding an existing plugin to 'required plugins' allows a plugin to access resources specified by another plugin. The plugin that shares resources becomes an required dependency to the client plugin. In this case, add 'edu.cs.usask.ca.silver.plugin.base(1.0.0)' which provides business logic API to 'required Plug-ins'. Thus, Java programs specified in the 'test' plugin now have the ability to access the business logic API.
- To package the 'test' plugin to the system, open 'plugin.xml' under project named 'edu.cs.usask.ca.silver.plugin.rcp'. Under dependencies tab, add 'test' plugin to the 'required Plug-ins', as shown in Figure 7.3.

Now the newly added test plugin is added to SILVER.

7.3 Effectiveness from the User's Perspective

It is important that the system provides users with adequate supports that allows them to perform the tasks that they desire. As part of the evaluation process, a user evaluation with a small group of five experienced modelers (with modeling experience from 6 months to 2 years) was performed. The group members were given the two sample model projects introduced in Chapter 6 on which to provides feedback with respect to the adequacy of the model hierarchy support, change awareness information and visualization provided by the system workspace. During the evaluation, the modelers were asked to perform collaborative modeling tasks on provided projects. The tasks involved information finding, model version creation and scenario analysis based on the progress that the group of modelers have made. Together the modelers performed distributed collaborative modeling without using other collaboration tools such as Email or FTP that otherwise would have been used without SILVER. After the evaluation, each modeler was asked to fill in a questionnaire to rate difficulty (1 as the least difficult, 6 as almost impossible) to perform change awareness related tasks during collaboration. Figure 7.4 presents the result from the evaluation questionnaire filled by the modelers immediately after the evaluation. The detailed SILVER evaluation questionnaire can be found in the Appendix B.

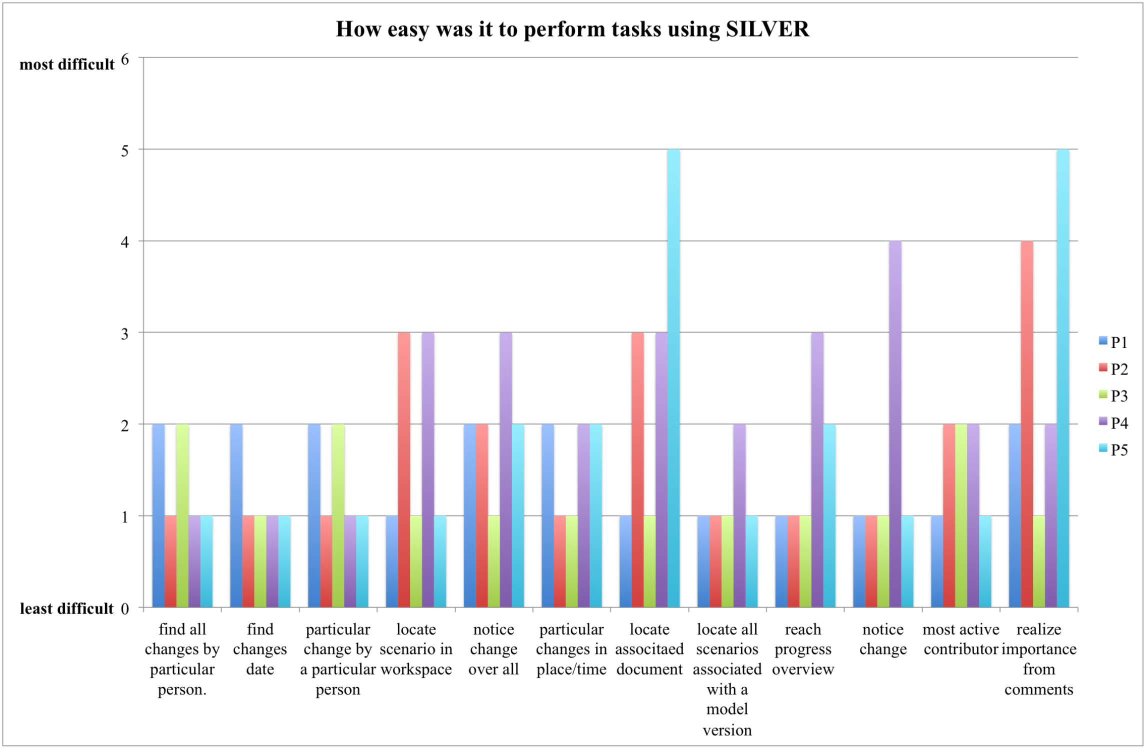


Figure 7.4: SILVER evaluation analysis

In general, all group members agreed that both modeling process hierarchy and change awareness information provided was beneficial to the modeling process in which they were participating, and that it made the modeling process in general more productive than if SILVER was not used.

Given that modeling process hierarchy and change awareness are different aspects in supporting the modeling process, the discussion for each evaluation is separated in two

Category	Information	Sample Question
Where	Location history	Which model version contains the latest scenario
	Edit history	What model versions has been changed yesterday?
Who	Identity	Who has created the model project?
	Authorship history	Who were participating in the model project?
What	Action history	What has been changed by a particular person yesterday?
When	Event history	When was the latest model version created in a project?
Why	Cognitive history	Why were a set of new scenarios added to a particular model version?
	Motivation history	Why did Person A add a new model version to a model project?

Table 7.1: Change Awareness Evaluation

sections below.

7.3.1 Effectiveness of Modeling Process Hierarchy

The evaluation of the modeling process hierarchy include modeling tasks that cover model buliding, analysis and simulation sharing. The modelers found the concept of modeling process hierarchy suitable for their day-to-day work. In particular they found much effort was reduced while saving successive versions of models, organizing simulation outputs and communicating updates regarding particular entities from the modeling process hierarchy. In spite of the tasks associated with the modeling process itself, the modelers requested additional features such as automation in running a set of simulations, the ability to create custom tags based on the content in the modeling process hierarchy and the ability to work with more types of models such as Monte Carlo and modeling software such as powersim, etc [8].

7.3.2 Effectiveness of Change Awareness

The evaluation of the awareness workspace include information finding tasks with respect to each section of change awareness. In particular, the effectiveness of the following awareness information were investigated. Table 7.1 gave sample question with respect to each change awareness category supported by SILVER:

While all modelers agreed that the change awareness information was useful in keeping the higher level overview of the modeling process, all users provided comments about how the change awarenes information from the awareness workspace could be accessed differently

One of the users felt that it was important to ensure consistency in the operation of the widgets and the visualization of the awareness information. This is an issue that will be focussed on more heavily in future work.

7.4 Limitations

This section will discuss the limitations of modeling with SILVER. The limitations are based on the feedbacks from modelers who have participated in SILVER evaluation. Many of the limitations discussed are due to the scope of the current project, and future development will attempt to address limitations of this nature.

- *Data synchronization between storage spaces.* Currently, SILVER lacks a data synchronization mechanism that allows users to synchronize all changes of a modeling project between local and remote storage. The only way for users to share changes in collaboration is to redo the changes on the other storage. Providing a synchronization mechanism requires identity of the updated entity as well as the data describing how an entity was updated. SILVER is currently able to provide both information by assigning an UUID to each entity in the modeling process hierarchy and to each change within the modeling process hierarchy at database level. UUIDs are 128 bit numbers which are intended to have a high likelihood of uniqueness over space and time. They can be locally generated in database without contacting a global registration authority [7]. However, developing the synchronization mechanism which employs techniques in data comparing, change scripting and data serialization over the network is beyond the scope of this project.

- *Immediate change notification modeling process hierarchy.* SILVER offers partial support to change awareness through awareness workspace. However, a user will not typically be aware of a change until they look up changes through the workspace. It would be useful if SILVER were proactive in notifying immediate changes in a near-to-real-time fashion.

- *Change history contains all changes across storage.* The capacity to display of specific changes in respect to particular entity within the modeling process hierarchy is not currently supported. This limits the user who is looking for information about a particular project or versions from abundant data that contains much unneeded information. Another solution would be adding an interactive search tool to the change history.

- *Incomplete support for modeling packages.* While popular, Vensim and Anylogic are a small sub set among modeling tools. With the current architecture, it is possible to add support to additional popular modeling tools such as PowerSim, Stella, Repast and NetLogo in the future [8, 10, 5].

- *No method of merging models* For example in System Dynamics modeling, modelers often pursue assumptions based on equations applied in the model. They require a tool that allows the system to compare the equation difference between models and the ability to merge them. Such a feature can be added by comparing different versions of model source files and then save away equation differences using SILVER Business Logic API.

- *Visual Workspace is Not Updated After Changes Have Been Applied* The activated awareness workspace could not be updated after changes have been applied to the modeling process hierarchy. This is due to the in-memory database constructed for visual workspace and ChangeSet history. However, the user can proactively maintain the awareness by closing and re-opening the visual workspace.

- *Import or Export Data* SILVER currently does not support exporting data to file on disk or re-constructing a database from external data. Such feature is useful in migrating database between computers or manually perform database backup. Especially given that most modelers may already have a large base of model artifacts, it would be valuable efficient to allow users quickly construct data from such existing outputs.

CHAPTER 8

CONCLUSION

This chapter concludes the thesis by providing a brief summary of the research problem, goals, and approach to the solution. The major and minor contributions of the thesis are restated. The final section of the chapter provides an introduction to the planned future work and directions for the research started in this thesis.

8.1 Summary of Research and Contributions

This thesis confronted the problem that *it is difficult to perform collaborative modeling*. Collaborative modeling is the modeling process performed by teams of modelers and other stakeholders who are spatially distributed. The difficulties exists because of the effort required in maintaining various modeling related artifacts and in being aware of overall progress of the project. This is mainly due to the lack of software that focuses on supporting the process of modeling other than manipulation of the artifact itself. The issues includes model versioning, scenario bookkeeping, documenting intentions, linking to external documentation, distributed communication, automated logging of user changes, and transforming the changes to meaningful updates on the GUIs of the remote users.

The main goal of the research was to provide a platform to reduce the effort in collaborative modeling. A second goal of the research was to provide an informative collaborative environment for modeling collaborators. The motivation is the notion that better support of change awareness and artifact management means better collaboration for modelers. The problem was addressed by building a software system that allows modelers to easily perform the iterative modeling process in a modeling team. The main focus of the system is the modeling process hierarchy, compatibility with external modeling software, and change awareness support. The information about "When" and "Who" in past changes is the key awareness information provided by SILVER.

SILVER simplifies the construction of groupware that supports group awareness in four ways:

- It provides a modeling process hierarchy to support a rigorous modeling process.
- It provides both distributed and centralized storage infrastructure.
- It provides support of a set of popular modeling software.
- It has visual workspace component that provides support of change awareness in collaborative modeling.

A number of steps were completed in the research:

- Specifying the modeling process hierarchy in terms of collaborative modeling
- Specifying modeling requirements in terms of change awareness
- Developing storage infrastructure
- Developing business logic infrastructure
- Developing GUI component infrastructure
- Developing the visual workspace infrastructure
- Developing support for the Anylogic and Vensim simulation packages.

The major contributions of the research are:

- A software system that reduces difficulty in collaborative modeling. The system includes mechanisms to create a model project from existing Anylogic or Vensim simulations and to create initial versions and baseline scenarios for modeling process hierarchy.
- A quicker way to create incremental version of models. The system allows a user to quickly reflect model structural changes by importing and reviewing revisions as roll-backs if necessary.
- An easier way to explore models by creating scenarios, changing parameter values and running simulations for results.

- Workspace components that provide support for change awareness.
- A mechanism that allows users to associate external files with particular entities in the modeling process hierarchy as additional documentation.

  A minor contribution of the research is:

- Controlling Anylogic simulation by employing AspectJ load-time weaving.

Before the research started, a prototype of SILVER using a different set of technologies were built by Yudi Xue, Daniel Funk and Mike Taylor as a class project. Due to the limitations of the technology, the prototype could never been built into binary executable form. The research inherited the basic data storage schema and the idea of component-based desktop software from the prototype. However, the actual code have been completely re-written.

Overall, given the summarized list of research steps and the list of contributions, SILVER reduced the efforts in distributed modeling process by providing support to MPH and to change awareness as a whole.

8.2 Future Work

Following the completion of the thesis, enhancements and extensions have been planned for a future release of the SILVER system. First, more testing and bug fixes will be performed on the existing system. Currently, SILVER contains strong unit tests coverage over the core storage components. However, from a software engineering perspective, SILVER lacks automated system tests and functional tests to ensure proper behaviors in business logic and GUI components. The components are error-prone and will be tested along future releases. Second, users felt reluctant to use SILVER for collaboration since they are not equipped with the requisite technical knowledge to set up a remote database. This problem can be solved by abstracting away the details of configuration by use of automation. Or, through SILVER development, a potential idea has been noted that a web-based hub might be useful for remote modeling collaboration. In this scheme, SILVER will be used as an desktop client which allows user to freely collaborate with other registered users on the hub and explore models. However, such idea require extensive resource and imply additional web-based technologies to make it possible. Third, more discussions and studies will occur regarding change awareness support. Information about all changes and actions a modeler made before reaching a goal can reflect how a modeler performs simulation modeling other than building the model. Such information regarding past actions may become useful for education purposes and enhancing confidence between the team and clients. Following this, more UI components will be developed to display particular change awareness information that is missing in certain modeling context. Fourth, SILVER lacks the ability to let users to re-use old scenarios in a new version of the model, it can be partially done by copying old scenarios to the new model version. However, if newer model version have removed certain parameters, SILVER will require a mechanism to verify that all scenario parameters are available in the new model version. Next, support to load-balanced simulation across a cluster would allow multiple simulations to be executed in a batch. It is possible to delegate SILVER's simulation runner to server cluster by further automating the simulation running process and by developing an adapter on SILVER to send scenario information to the server cluster. Such infrastructure require further investigation on simulation control over multiple packages as well as determining cluster architecture. These tasks are outside the scope of this thesis. Finally, all other components that are discovered through discussion and evaluation will be created. All newly created components will be followed by more testing.

Support of data synchronization between storage will be added by scripts that will compare copies of a particular model project stored in different locations and execute necessary action for the purpose of synchronization. Also, the capability to run simulations on additional modeling packages will be investigated. The re-usability of the business logic component will be tested by integrating various modeling packages that requires different forms of support.

References

[1] Anylogic — XJ Technologies Inc. http://www.xjtek.com accessed Dec, 2011.

[2] Apache commons pool: A mini-framework to correctly implement object pooling in java. http://commons.apache.org/pool/ accessed Dec, 2011.

[3] The base16, base32, and base64 data encodings. http://tools.ietf.org/html/rfc4648, accessed Dec, 2011.

[4] Eclipse Rich Client Platform IDE. http://www.eclipse.org/downloads/packages/eclipse-rcp-and-rap-developers/indigosr2, accessed Apr, 2012.

[5] Netlogo homepage. http://ccl.northwestern.edu/netlogo/ accessed Dec, 2011.

[6] OSGi Alliance — Specifications. http://www.osgi.org/Specifications/HomePage accessed Dec, 2011.

[7] OSSP: OSSP uuid. http://www.ossp.org/pkg/lib/uuid/ accessed Dec, 2011.

[8] Powersim studio 8 — powersim software. http://www.powersim.com/ accessed Dec, 2011.

[9] Silver. http://code.google.com/p/uofs-silver-hg, accessed April, 2012.

[10] Stella modeling & simulation software. http://www.iseesystems.com accessed Dec, 2011.

[11] Vensim — Ventana Systems, Inc. www.vensim.com accessed Dec, 2011, Dec 2011.

[12] Y. Borodin, J. Bigham, R. Raman, and I. V. Ramakrishnan. What's new?: making web page updates accessible. In *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, Assets '08, pages 145–152, New York, NY, USA, 2008. ACM.

[13] D. Chamberlin and R. Boyce. Sequel: A structured english query language. In *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control*, SIGFIDET '74, pages 249–264, New York, NY, USA, 1974. ACM.

[14] H. H. Clark and S. Brennan. Grounding in communication. pages 127–149, 1991.

[15] S. Clemens. *Component Software*. New York : ACM Press ; Harlow, England ; Reading, Mass. : Addison-Wesley, 1997.

[16] C. Collins, G. Penn, and S. Carpendale. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15:1009–1016, November 2009.

[17] P. Dourish and S. Bly. Portholes: supporting awareness in a distributed work group. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '92, pages 541–547, New York, NY, USA, 1992. ACM.

[18] A. Eisenberg, J. Melton, K. Kulkarni, J. Michels, and F. Zemke. Sql:2003 has been published. *SIGMOD Rec.*, 33:119–126, March 2004.

[19] M. Endsley. Measurement of situation awareness in dynamic systems. *Human Factors The Journal of the Human Factors and Ergonomics Society*, 37(1):65–84, 1995.

[20] J. Forrester. *Industrial Dynamics*. Pegasus Communications, 1961.

[21] J. Forrester. *Urban Dynamics*. Pegasus Communications, 1969.

[22] Apache Foundation. Apache Derby. http://db.apache.org/derby/ accessed Dec, 2011.

[23] Eclipse Foundation. Eclipse - the eclipse foundation open source community. http://www.eclipse.org/ accessed Dec, 2011.

[24] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.

[25] G. W. Furnas. Readings in information visualization. chapter The FISHEYE view: a new look at structured files, pages 312–330. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[26] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[27] J. Grey and A. Reuter. *Transaction processing: concepts and techniques*. Morgan K., September 1992.

[28] C. Gutwin. *Workspace Awareness in Real-Time Distributed Groupware*. PhD thesis, Department of Computer Science, University of Calgary, 1997.

[29] J. Heer, S. Card, and J. Landay. prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '05, pages 421–430, New York, NY, USA, 2005. ACM.

[30] W. Hill, J. Hollan, D. Wroblewski, and T. McCandless. Edit wear and read wear. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '92, pages 3–9, New York, NY, USA, 1992. ACM.

[31] J. Homer. Why we iterate: scientific modeling in theory and practice. *System Dynamics Review*, 12(1):1–19, 1996.

[32] J.W. Hunt and M.D. Mcllroy. An algorithm for differential file comparison. Computing Science Technical Report 41, Bell Laboratories, 1975.

[33] T. Ioannis, B. Giuseppe, E. Peter, and T. Roberto. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[34] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. Griswold. An overview of aspectj. In *Proceedings of the 15th European Conference on Object-Oriented Programming*, ECOOP '01, pages 327–353, London, UK, UK, 2001. Springer-Verlag.

[35] D. Kurlander. Graphical editing by example (abstract). In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, CHI '93, pages 529–, New York, NY, USA, 1993. ACM.

[36] T. Lindholm and F. Yellin. *The Java^{TM} Virtual Machine Specification*. Sun, April 1999.

[37] R. Marc. The source code control system. *IEEE Transition*, Software Engineering SE-1,4:364–70, Dec 1975.

[38] N. Mark. Xml. *Serials Review*, 25(1):117 – 121, 1999.

[39] J. McAffer, J. Lemieux, and C. Aniszczyk. *Eclipse Rich Client Platform*. Addison-Wesley Professional, 2nd edition, May 2010.

[40] J. McAffer, P. Vanderlei, and S. Archer. *OSGi and Equinox: Creating Highly Modular Java Systems*. Addison-Wesley Professional, Feb 2010.

[41] K. Michael and J. Prashant. Pooling pattern. In *EuroPLoP*, pages 497–510, 2002.

[42] Sun Developer Network. Java applet. http://java.sun.com/applets/ accessed Dec, 2011.

[43] G. Nigel. *Agent-Based Models (Quantitative Applications in the Social Sciences).* SAGE publications, 2007.

[44] Jr. Nunamaker, F. Jay, R. Briggs, D. Mittleman, D. Vogel, and P. Balthazard. Lessons from a dozen years of group support systems research: a discussion of lab and field findings. *J. Manage. Inf. Syst.*, 13:163–207, December 1996.

[45] G. Olson and J. Olson. Distance matters. *Hum.-Comput. Interact.*, 15:139–178, September 2000.

[46] Oracle. Java se technologies - database. http://www.oracle.com/technetwork/java/javase/jdbc/index.html accessed Dec, 2011.

[47] Oracle. Java transaction api (jta). http://www.oracle.com/technetwork/java/javaee/jta/index.html accessed Dec, 2011.

[48] N. Osgood. Silver: Software in support of the system dynamics modeling process. *ICSD*, 2009.

[49] S. Railsback and V. Grimm. *Agent-based and Individual-based Modeling: A Practical Introduction.* Princeton University Press, 2011.

[50] L. Ramnivas. *AspectJ in Action.* Manning Publications, July 2003.

[51] J. Randers. *Conceptualizing Dynamic Models of Social Systems: Lessons from a Study of Social Change.* PhD thesis, MIT, 1973.

[52] J. Randers. *Guidelines for Model Conceptualization. In Elements of the System Dynamics Method, ed. J. Randers.* MIT Press., 1980.

[53] Forio Online Simulations. Forio online simulations. http://forio.com/ accessed Dec, 2011.

[54] M. Sparling. Lessons learned through six years of component-based development. *Commun. ACM*, 43:47–53, October 2000.

[55] J. Sterman. *Business Dynamics: Systems Thinking and Modeling for a Complex World.* McGraw Hill, 2000.

[56] J. Tam and S. Greenberg. A framework for asynchronous change awareness in collaborative documents and workspaces. *Int. J. Hum.-Comput. Stud.*, 64:583–598, July 2006.

[57] A. Tavares and M.T. Valente. A gentle introduction to OSGi. *SIGSOFT Softw. Eng. Notes*, 33:8:1–8:5, August 2008.

[58] J. Vennix. *Group model building: facilitating team learning using system dynamics.* Wiley, 1996.

APPENDIX A

DATABASE ER DIAGRAM

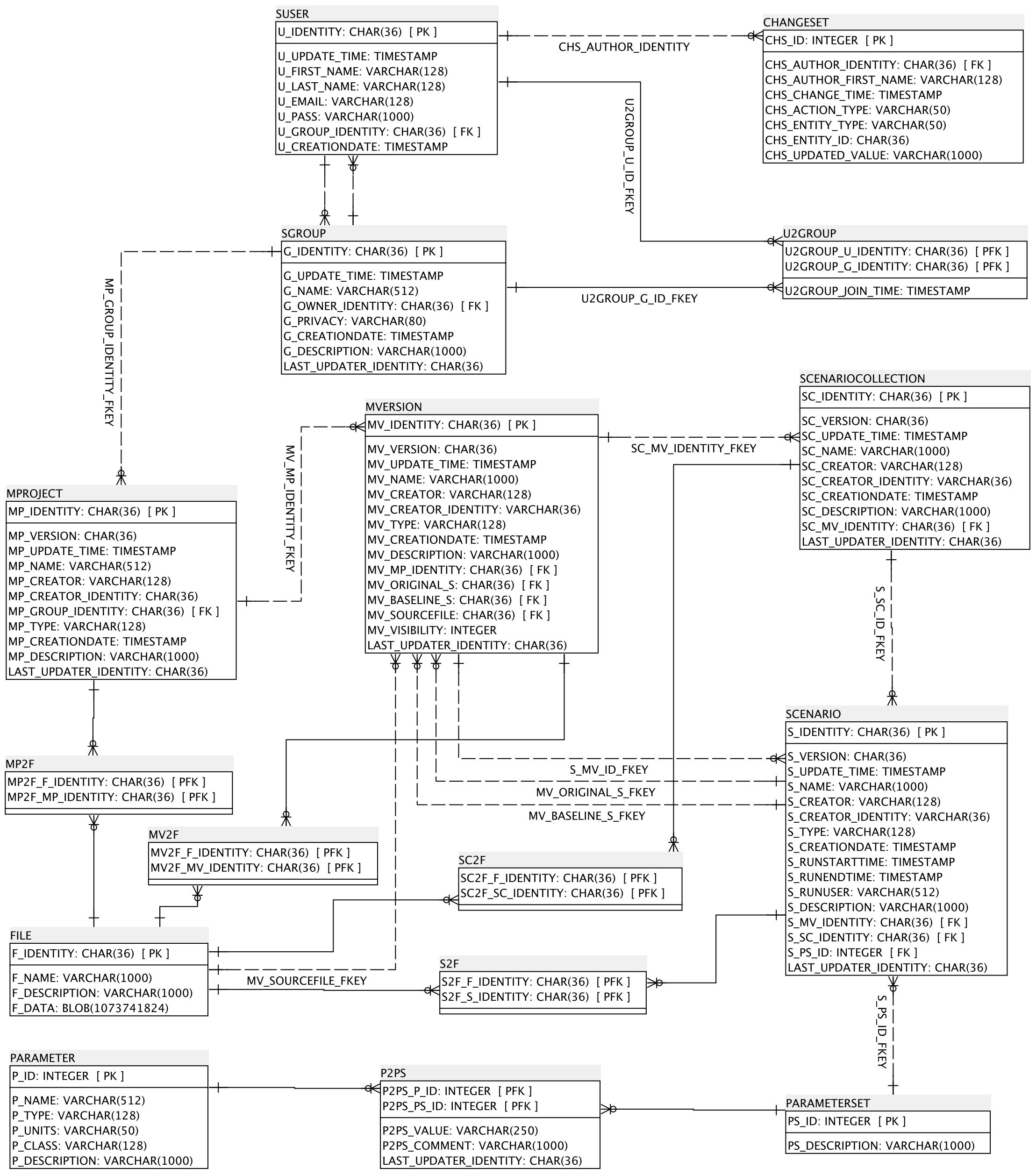

SUSER
U_IDENTITY: CHAR(36) [PK]
U_UPDATE_TIME: TIMESTAMP
U_FIRST_NAME: VARCHAR(128)
U_LAST_NAME: VARCHAR(128)
U_EMAIL: VARCHAR(128)
U_PASS: VARCHAR(1000)
U_GROUP_IDENTITY: CHAR(36) [FK]
U_CREATIONDATE: TIMESTAMP
CHS_AUTHOR_IDENTITY
CHANGESET
CHS_ID: INTEGER [PK]
CHS_AUTHOR_IDENTITY: CHAR(36) [FK]
CHS_AUTHOR_FIRST_NAME: VARCHAR(128)
CHS_CHANGE_TIME: TIMESTAMP
CHS_ACTION_TYPE: VARCHAR(50)
CHS_ENTITY_TYPE: VARCHAR(50)
CHS_ENTITY_ID: CHAR(36)
CHS_UPDATED_VALUE: VARCHAR(1000)
U2GROUP_U_ID_FKEY
SGROUP
G_IDENTITY: CHAR(36) [PK]
G_UPDATE_TIME: TIMESTAMP
G_NAME: VARCHAR(512)
G_OWNER_IDENTITY: CHAR(36) [FK]
G_PRIVACY: VARCHAR(80)
G_CREATIONDATE: TIMESTAMP
G_DESCRIPTION: VARCHAR(1000)
LAST_UPDATER_IDENTITY: CHAR(36)
U2GROUP
U2GROUP_U_IDENTITY: CHAR(36) [PFK]
U2GROUP_G_IDENTITY: CHAR(36) [PFK]
U2GROUP_JOIN_TIME: TIMESTAMP
U2GROUP_G_ID_FKEY
MP_GROUP_IDENTITY_FKEY
SCENARIOCOLLECTION
SC_IDENTITY: CHAR(36) [PK]
SC_VERSION: CHAR(36)
SC_UPDATE_TIME: TIMESTAMP
SC_NAME: VARCHAR(1000)
SC_CREATOR: VARCHAR(128)
SC_CREATOR_IDENTITY: VARCHAR(36)
SC_CREATIONDATE: TIMESTAMP
SC_DESCRIPTION: VARCHAR(1000)
SC_MV_IDENTITY: CHAR(36) [FK]
LAST_UPDATER_IDENTITY: CHAR(36)
MVERSION
MV_IDENTITY: CHAR(36) [PK]
MV_VERSION: CHAR(36)
MV_UPDATE_TIME: TIMESTAMP
MV_NAME: VARCHAR(1000)
MV_CREATOR: VARCHAR(128)
MV_CREATOR_IDENTITY: VARCHAR(36)
MV_TYPE: VARCHAR(128)
MV_CREATIONDATE: TIMESTAMP
MV_DESCRIPTION: VARCHAR(1000)
MV_MP_IDENTITY: CHAR(36) [FK]
MV_ORIGINAL_S: CHAR(36) [FK]
MV_BASELINE_S: CHAR(36) [FK]
MV_SOURCEFILE: CHAR(36) [FK]
MV_VISIBILITY: INTEGER
LAST_UPDATER_IDENTITY: CHAR(36)
SC_MV_IDENTITY_FKEY
MV_MP_IDENTITY_FKEY
MPROJECT
MP_IDENTITY: CHAR(36) [PK]
MP_VERSION: CHAR(36)
MP_UPDATE_TIME: TIMESTAMP
MP_NAME: VARCHAR(512)
MP_CREATOR: VARCHAR(128)
MP_CREATOR_IDENTITY: CHAR(36)
MP_GROUP_IDENTITY: CHAR(36) [FK]
MP_TYPE: VARCHAR(128)
MP_CREATIONDATE: TIMESTAMP
MP_DESCRIPTION: VARCHAR(1000)
LAST_UPDATER_IDENTITY: CHAR(36)
S_SC_ID_FKEY
SCENARIO
S_IDENTITY: CHAR(36) [PK]
S_VERSION: CHAR(36)
S_UPDATE_TIME: TIMESTAMP
S_NAME: VARCHAR(1000)
S_CREATOR: VARCHAR(128)
S_CREATOR_IDENTITY: VARCHAR(36)
S_TYPE: VARCHAR(128)
S_CREATIONDATE: TIMESTAMP
S_RUNSTARTTIME: TIMESTAMP
S_RUNENDTIME: TIMESTAMP
S_RUNUSER: VARCHAR(512)
S_DESCRIPTION: VARCHAR(1000)
S_MV_IDENTITY: CHAR(36) [FK]
S_SC_IDENTITY: CHAR(36) [FK]
S_PS_ID: INTEGER [FK]
LAST_UPDATER_IDENTITY: CHAR(36)
S_MV_ID_FKEY
MV_ORIGINAL_S_FKEY
MV_BASELINE_S_FKEY
MP2F
MP2F_F_IDENTITY: CHAR(36) [PFK]
MP2F_MP_IDENTITY: CHAR(36) [PFK]
MV2F
MV2F_F_IDENTITY: CHAR(36) [PFK]
MV2F_MV_IDENTITY: CHAR(36) [PFK]
SC2F
SC2F_F_IDENTITY: CHAR(36) [PFK]
SC2F_SC_IDENTITY: CHAR(36) [PFK]
FILE
F_IDENTITY: CHAR(36) [PK]
F_NAME: VARCHAR(1000)
F_DESCRIPTION: VARCHAR(1000)
F_DATA: BLOB(1073741824)
MV_SOURCEFILE_FKEY
S2F
S2F_F_IDENTITY: CHAR(36) [PFK]
S2F_S_IDENTITY: CHAR(36) [PFK]
S_PS_ID_FKEY
PARAMETER
P_ID: INTEGER [PK]
P_NAME: VARCHAR(512)
P_TYPE: VARCHAR(128)
P_UNITS: VARCHAR(50)
P_CLASS: VARCHAR(128)
P_DESCRIPTION: VARCHAR(1000)
P2PS
P2PS_P_ID: INTEGER [PFK]
P2PS_PS_ID: INTEGER [PFK]
P2PS_VALUE: VARCHAR(250)
P2PS_COMMENT: VARCHAR(1000)
LAST_UPDATER_IDENTITY: CHAR(36)
PARAMETERSET
PS_ID: INTEGER [PK]
PS_DESCRIPTION: VARCHAR(1000)

Figure A.1: Database Entity Relationship

APPENDIX B

SILVER EVALUATION FORM

Thank you for helping in SILVER evaluation! You have already gone through two procedures. Please answer each question in the following questionnaire.

* Required

Participant ID *

Where are you located? * please specify your city and province

How long have you been involved in modeling process? * includes learning and using Vensim or Anylogic software

never less than a month 1 - 6 months 6 months - 1 year 1 year - 2 years 2 years and longer

How many modeling projects were you involved in collaboration? * Please give a rough estimation about the number of projects

What is the average length of the modeling projects you have participated? *

2 weeks 2 weeks - 1 month 1 - 2 months 3 - 6 months 6 months - 1 year longer than 1 year

How many modeling project you were involved in parallel at most? * give the largest number of projects that you have been involved at the same time

How difficult to understand the concept of modeling process collaboration model implemented in SILVER? *

1 2 3 4 5

very easy very difficult

How difficult it is to look up changes done by a particular person. *

1 2 3 4 5 very easy very difficult

How difficult it is to look up all changes associated for a particular date (for example yesterday)? *

1 2 3 4 5 very easy very difficult

How difficult it is to find a particular model version or scenario created by particular person? *

1 2 3 4 5 very easy very difficult

How difficult it is to locate a scenario without the knowledge of which version the scenario might be associated with? *

1 2 3 4 5 very easy very difficult

How difficult it is to find what have been changed overall (across multiple projects) in a certain period of time? *

1 2 3 4 5 very easy very difficult

How difficult it is to find what have been changed within a particular model project within a certain period of time? *

1 2 3 4 5 very easy very difficult

How difficult it is to locate a certain document attached by a person without knowing where the document is located? *

1 2 3 4 5 very easy very difficult

How difficult it is to locate all scenarios associated to a certain version of model? *

1 2 3 4 5 very easy very difficult

How difficult it is for you to reach an overview about the progress of the modeling process? *

1 2 3 4 5 very easy very difficult

How difficult it is to notice that a model project has been updated? *

1 2 3 4 5 very easy very difficult

Who is the most active contributor in the modeling process? (a particular project) *

1 2 3 4 5 very easy very difficult

Did you come across comments from scenario that reflects suggestions that is important? *

1 2 3 4 5 very easy very difficult

How do you think about the overall network/performance quality *

1 2 3 4 5 very easy very difficult

Additional comments on functionalities

Powered by Google Docs Report Abuse - Terms of Service - Additional Terms