# SILVER: Software in Support of the System Dynamics Modeling Process

**Nathaniel Osgood**
Department of Computer Science
University of Saskatchewan
osgood@cs.usask.ca

**Abstract**: While the System Dynamics modeling process can yield invaluable high level insights, it gives rise to a tremendous amount of detail complexity. In the course of their work, modelers must track successive model versions, the motivation for and assumptions underlying particular "what if" scenarios, and the implicit relationships between scenarios, model versions and various external artifacts such as spreadsheets, symbolic mathematics calculations, and external documentation. Failure to adequately manage such complexity can reduce the transparency, reliability, and credibility of the modeling process. While adherence to good modeling practices can aid this process, it often falls prey to corner-cutting or human error. This paper describes software that helps manage such complexity, by permitting modelers to easily access and succinctly compare historic versions of a model, by making explicit linkages between scenarios, the model versions and assumptions underlying them, and the motivations for and external files associated with model artifacts.

## 1 Introduction

The System Dynamics modeling process offers great value in allowing stakeholders to rise above the welter of operational detail and perform long-term strategic planning. There is a certain irony in the fact that important higher-level insights can emerge from a modeling process that frequently generates and requires management of a massive amount of day-to-day detail complexity. In the course of their work, for example, modelers must keep track of successive refinements and variations on models, the sets of "what if" scenarios already explored, and the relationships between those scenarios and multiple other entities: The versions of models that were used to create them, the particular assumptions that underlay the scenarios, the intensions underlying those choice of assumptions, insights gained from those scenarios, and often implicit links to external artifacts that aided in determination of parameter values or that further analyze scenario results.

Most System Dynamics modelers have adopted conventions and practices to help assist with the bookkeeping complexity associated with modeling: Good modeling practice suggests that a model be accompanied by clear external documentation regarding scenarios investigated and their underlying assumptions and motivations, the use of descriptive scenario filenames, carefully flagging of variables that must be reset to baseline values after investigating alternative scenarios, and the incorporation of explicit model version numbers via a model parameter. While such conventions do aid modelers in grappling with the detail complexity of the modeling process, reliance on such bookkeeping conventions relies heavily on continued modeler commitment, known all too well to be fallible, particularly under time pressure. It is thus all too easy for a modeler in a rush to inadvertently overwrite an earlier version of model, complicating or preventing the recreation of a set of earlier scenario results. Even putting corner-cutting aside, human error is all too likely, and can lead to mistaken interpretation of model results. Consider, for example, cases in which a modeler forgets to re-run certain model scenarios following the correction of an important structural error (and misattributes results from scenarios produced through earlier simulations to the corrected version of the model), neglects to reset to its baseline value a parameter following a scenario run (and thereby misconstrues the assumptions involved in subsequently explored model scenarios), or deletes a given model version that has retrospectively been found to be in error (but neglects to delete associated scenario results, which are later erroneously assumed to be legitimate).

Even ideal adherence to good modeling practices carries some overhead. With the longer times between simulations required when manually keeping track of the relationships between and documentation for model artifacts, fewer earlier results are kept in short-term memory, the learning process can be slowed, and the modeler places themselves at risk of "seeing the trees but not the forest". In the face of lapses in the manual bookkeeping process, the transparency of the modeling process may also be adversely impacted, with the modelers themselves facing confusion and difficulties interpreting and recreating previously-produced modeling results.

Finally, the current fragmentation of information related to model artifacts amongst multiple documents also raises barriers to effective collaboration: Even where information establishing the relationship between multiple artifacts is carefully maintained, access to such information – and the artifacts themselves – amongst stakeholders will frequently not be universal or complete. The complexity of the modeling bookkeeping process can itself discourage close stakeholder involvement in the modeling process, and raise concerns about the reliability and integrity of the modeling process.

Within this paper we describe the functionality of soon-to-be-released open-source software (SImuLation VERsioning, or SILVER) that seeks to reduce the bookkeeping and version-control overhead associated with models, and to lower the barriers to effective and transparent collaboration. While still in its initial stages of functionality, the software will support modeling projects built under a wide variety of popular modeling platforms, and will work both as a stand-alone desktop package and in a firewall-transparent network configuration.

The remainder of this paper is organized as follows: Within the next section, we establish terminological conventions used to describe common modeling practices. This terminology is then used throughout the remainder of the paper and is captured in the software design and user interface. We then turn in Section 3 to expand upon the above discussion of important gaps in support for the modeling practice and survey the implications of such gaps for the quality and efficiency of the modeling process. Within Section 4, we describe the high-level functionality of the software. Within the final section, we note gaps within the software, and comment on some of the challenges & barriers encountered.

## 2 Domain Terminology

In spite of underlying similarities uniting System Dynamics modeling projects, the terminology used to describe aspects of the modeling process has not been standardized. In order to reduce the risk of reader/user confusion, we seek in this paper to make use of a consistent set of terminology when describing the modeling process. This section of the paper introduces our terminological conventions by way of describing the conceptual model of the modeling process supported by the software.

For the sake of this paper and software support, we view modeling as taking place within the context of *modeling projects*, with each such modeling project including one or more threads of work that producing a set of ongoing set of modeling artifacts.

The work done as part of a given modeling project will give rise to one or more versions of the model document – what we term *model versions*. We use the term *model version* to refer to a model with a specific static structure, treat any change to that structure as creating a new model version. Frequently such versions are produced in a chronological order (in a process of successive refinement), but sometimes additional structures are used – for example, a series of exploratory models may be built to e.g. investigate the impact of particular approaches to disaggregation.

While the structure of a given model version is static, it will frequently be used to investigate many *scenarios*, with each scenario investigating how a specific model version (presumably an approximation to some real-world system) behaves under a particular set of conditions – for example, with particular policies in place or under a specific set of assumptions regarding exogenous factors. The software assumes that each such scenario is realized by simulating the model with a particular assignment of values to each of the model *parameters* – a construct we term a *parameter set*. The associated *scenario* is defined by the pairing of a specific model version with a particular parameter set. For many simulation packages, the results of the simulation will be captured in a *dataset* file reporting the trajectory of model variables and parameters over time. Generally, each assignment of value of a parameter in such a parameter set, as well as the parameter set as a whole, have a specific motivation underlying them, as would the associated scenario (the intension for pairing this parameter set with the particular model version at hand). A given parameter set is often formulated as part of a coherently defined *parameter set collection* examining successive variations of a set of assumptions. Frequently a hierarchy of such parameter set collections will exist. Commonly, one scenario for each model version will represent a *baseline* (reference) scenario, with the output from other scenarios being compared to that. It is a common practice to reuse parameter sets defined for earlier versions of the model for use with later versions of a model – for example, maintaining (but potentially elaborating) the parameter set associated with the baseline scenario as successive model versions evolve to add or change parameter values.

A modeler will often maintain *associated external files* (e.g. files in spreadsheets such as Microsoft Excel (Microsoft Corporation, 2007), symbolic mathematics packages such as Maple (Maplesoft, 2008) or Mathematica (Wolfram Research, 2008), or word processing documents), papers from the secondary literature associated with parameter sets, parameter set collections, or scenarios. Such documents may, for example, provide or document parameter values for a model, comment on and qualitatively interpret model outputs, or post-process model outputs (providing further analysis or more sophisticated charting or visualization of scenario output).

A modeler will frequently compare the results of multiple scenarios, or compare results of a scenario undertaken with one model version with results of corresponding scenarios associated with other model versions.

The modeling process described here is commonly undertaken in a team context. In many cases, the various parties will be in physically distinct locations. Even when the parties are located in the same institution and meet frequently, there is often a desire for the various parties to individually explore artifacts associated with the model – motivations for and results of scenarios, the model structure, parameters used for model output, analysis of scenario outputs, etc.

## 3 Gaps in Support of Today's Practices

While the exact degree of support provided for the modeling process described above varies amongst particular modeling packages, many aspects are partly or entirely unsupported by software. A few of the major gaps are as below:

- **Model Version Control**. Version control applications allow those creating software artifacts to store away successive versions of an artifact, retrieve earlier versions of an artifact, succinctly compare one version of an artifact to another, and exclusively "check-out" (and subsequently "check-in") artifacts for modification. In contrast to the extensive use of version control seen in the creation of other software artifacts, the use of version control is uncommon in the System Dynamics modeling community. Amongst less experienced modelers, this can easily result in an inability retrieve earlier versions of a model, or previously used versions of external files deriving parameter values.

- **Inefficient and Error-Prone Scenario Bookkeeping.** Important conceptual relationships exist between model versions, model parameters, and the scenarios simulated – a scenario and its output are specific to a model version and a particular parameter set. All too often, this relationship is merely implicit. Many modeling packages capture parameter values and model information in dataset files, but in the presence of reuse of model names, such information may not uniquely identify the model version; full parameter information may also be lacking for important classes of simulations (e.g. sensitivity analyses). Rigorous modeling practice will often maintain model scenario documentation manually, for example, in external word processing or spreadsheet documents. However, absent an overarching framework to preserve the referential integrity of the link between scenarios and their model versions and parameter sets, the connection can easily be broken – for example, by neglecting to create the external documentation for a given file, or by inadvertently deleting a model version without recognizing an important reference in the documentation that indicates dependency of many scenarios on that model version. Another pathology can occur when a model version used to create scenario output of recognized importance is updated and overridden, thereby preventing direct recreation of that output; worse yet, this may occur without the modeler recognizing it, thus leading to an error when attempting to recreate that output. A different type of problem can occur when a modeler fails to delete scenario output files that may in fact be deleted. This can occur out of out of the (perhaps grounded) fear that such results cannot be easily recreated, or out of an ungrounded fear that, somewhere, a piece of external documentation refers to such files and that they must be preserved to maintain the referential integrity of this link.

- **Unenforced Links to External Supporting Artifacts.** The supporting artifacts for the modeling process – in the form of external calculations used for parameter values, external time series files, or calculations that process outputs from one or more scenarios – often provide information necessary to reproduce model scenario results. There are thus important (but all too frequently implicit) relationships between such external files and the associated scenarios. Just as occurs for the links between modeling artifacts, the absence of mechanisms to enforce the referential integrity of such relationships risks their violation by the overwriting of one component (with or – worse – without the knowledge of the modeler), by deletion of one component of the link without the logical deletion of the other.

- **Lack of Recorded Modeling Intention.** A clear understanding of the motivation underlying the definition of a model version, model scenario, or parameter set is often of critical importance in interpreting the simulation results associated with those artifacts. While good modeling practice provides for external documentation of the intended motivation for and interpretation of successive model versions or model scenarios, and the lack of proximity of the documentation to the models frequently also lowers the transparency and efficiency of the modeling process. As a result of all of these factors, interpretation of model versions and results can become considerably less clear. Lack of transparent intention can also complicate the reuse of earlier groups of scenarios by forcing the modeler to rediscover or recall the intention behind and the definition of a collection of scenarios in order to adapt them to a new version of the model.

- **Absence of High-Level Scenario Structure.** While most scenarios are defined in the context of a family of scenarios, this contextual information is frequently not explicitly documented, and is merely implicit in the parameter values associated with those scenarios. This lack of transparency regarding context inhibits systematic exploration of policy or assumption space (by making it less clear which scenarios have in fact been simulated), lowers transparency (by facing a modeler with an exhaustive enumeration of scenarios that does not reflect the conceptual structure of parameter space), and can lead to needless clutter (by discouraging the deletion of a given scenario, out of concern for the fact that it may in fact be logically grouped with a set of scenarios that are being retained).

- **Barriers to Collaboration.** Most medium- and large-scale system dynamics models are built by diverse teams. An important need within team projects is the need to share understanding and insights. At present, most artifacts associated with the modeling process are either not shared at all, or are shared with remote parties via email or downloads from a website. While email and downloads do provide the diverse parties some degree of access to models and model results, it fails to provide a higher-level picture regarding the status of the modeling process (e.g. the set of scenarios run), and leads to the proliferation of a confusingly similar set of variants of a given artifact (e.g. similar model versions, or many scenarios documented only by their name and implicitly by parameter values).

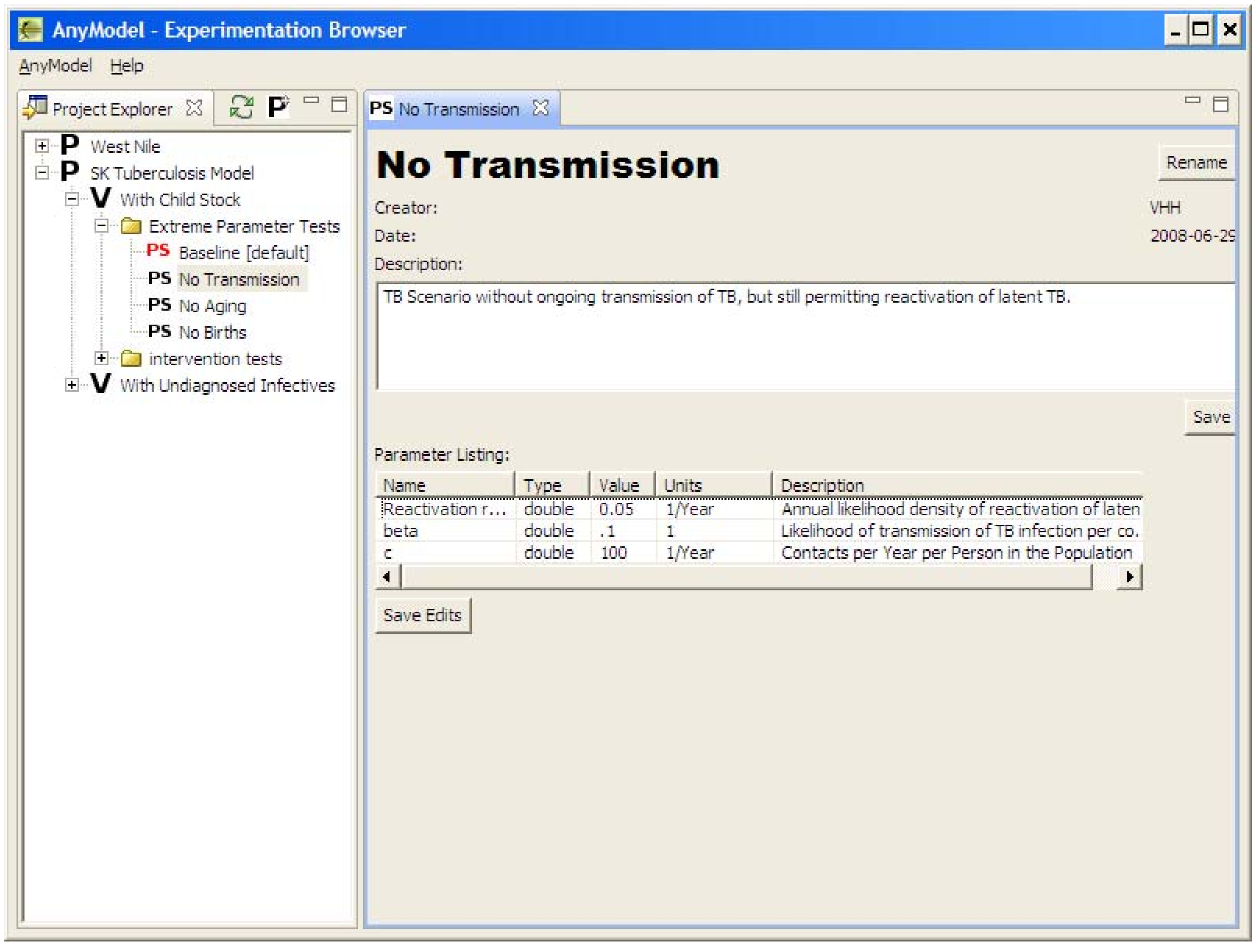
The gaps described above have substantive implications not only for the efficiency of the modeling process, but also for its correctness, for the reproducibility and transparency of the results, and for stakeholder confidence in and ability to learn from the modeling process. Within the next section, we describe a software system that seeks to narrow many of these gaps.

## 4 Software Functionality

This section describes the functionality of the software package designed to support the modeling process that was described in Section 2, and which specifically seeks to address many of the gaps described in the previous section. This section first introduces a brief description of the common patterns of use of the software. We then turn to discuss collaboration with the software, and finally provide a brief overview of the software implementation.

### 4.1 Software Use

The user of the software is presented with a "master-detail" display such as that shown in Figure 1. The hierarchy on the left provides a structured means of exploring information related to a modeling project at a variety of levels of granularity. The user interface to access this information is grouped around various modeling concepts, described in Section 2. The major concepts included in this visual hierarchy are listed below. For each such concept, we list the motivation for representing it, the functionality that is supported for this class of information, and related information that is maintained.



- P West Nile
- P SK Tuberculosis Model
  - V With Child Stock
    - Extreme Parameter Tests
      - PS Baseline [default]
      - PS No Transmission
      - PS No Aging
      - PS No Births
    - intervention tests
  - V With Undiagnosed Infectives

**No Transmission**

| | |
|---|---|
| Creator: | VHH |
| Date: | 2008-06-29 |
| Description: | TB Scenario without ongoing transmission of TB, but still permitting reactivation of latent TB. |

Parameter Listing:

| Name | Type | Value | Units | Description |
|---|---|---|---|---|
| Reactivation r... | double | 0.05 | 1/Year | Annual likelihood density of reactivation of laten |
| beta | double | .1 | 1 | Likelihood of transmission of TB infection per co. |
| c | double | 100 | 1/Year | Contacts per Year per Person in the Population |

**Figure 1: Master Detail Form Used by SILVER**

### 4.1.1 Model Project

A model project collects information and software artifacts related to a line of modeling work. Information maintained includes the project title, project description, project creator, project contributors, date of creation, and the simulation modeling package associated with the project. At present, Vensim (Ventana Systems, 2004) and AnyLogic (XJ Technologies, 2007) are the simulation platforms for which SILVER provides fully automated driving of simulations; other packages may currently be used but will require manual user simulation of models. We anticipate providing SILVER with an extensible “plug in” interface in the near future that will enable software developers to allow additional modeling packages to work with the existing SILVER software for fully automatic driving of a simulation without any need to reinstall SILVER or modify the SILVER source code.

While projects are by default visible to any individual who wishes to browse the associated repository, the user is also given the option to designate the project as private. With the exception of the project creator modeling platform, and creation date (which are fixed at time of creation of the project), the information may be updated on an ongoing basis.

### 4.1.2 Model Version

A model version object in the interface hierarchy captures information related to an instance of a model with a specific structure. A user may specify additional external files to be maintained with a model version. Examples might be documentation related to model structure, exogenous datasets, etc.

It is important to recognize that a model version is structurally immutable. While assignments of different constants to model parameters may be freely made within the scope of a model version, any structural modifications to a model must be checked in as a new model version. SILVER does not recognize any notion of a "current model version" – All model versions associated with a modeling project are available for access by the user at any time through the model hierarchy.

Some pieces of information associated with a model are set at the time of definition of the model version, and may not be changed thereafter. Immutable information includes the identity of the creator, the time of creation, a unique version number, and the complete model document. Information that the user may update include model metadata such as the model title, a description, associated external files, and an indication as to whether this model version is private to the creator, or is to be shared with others.

The most important function supported for a model version is reconstitution of a model representation. A user may at any point request the software to recreate the model definition files for a given model version on their local system. For Vensim, for example, the information reconstituted would be the .mdl file; for AnyLogic, the .alp file. The model file can then be opened and manipulated by the user. If the user structurally modifies the model, it may be added to the system as a different model version, or deleted with knowledge that the original copy of that model version remains available in the SILVER software. For convenience, the software will also give the user the option of recreating related files in the same folder. Additional files that were reconstituted may also be updated and (re-) checked-in to the software by the user.

The software will also include functionality that provides a (generally succinct) summary of the structural changes between two versions of the model. Finally, the software allows for copying a model version into a different modeling project, which allows for "branching" of modeling efforts, in which one project may spin off several additional projects, which may have related but distinct aims.

### 4.1.3 Scenario Collections and Hierarchy

Each model version within the interface hierarchy is associated with a hierarchy of zero or more parameter sets. This hierarchy includes both scenario collections and specific scenarios and allows a user to group together scenarios into a tree-structured hierarchy. For example, a scenario collection might group scenarios that are evaluations of the system response to various interventions, while another may be created to test the model robustness with respect to various external shocks, or to examine model behavior in response to extreme parameter values. Each scenario collection is associated with a description, an indication as to whether it is private or public, and links to any external files (e.g. documentation of the motivations for the choice of the parameter space that has been established, or comments interpreting the results of running the model version on that space).

### 4.1.4 Scenarios & Parameter Sets

Each scenario represents the application of a model version under a particular set of “what if” assumptions. The scenario interface object maintains information on the description of the scenario (permitting the modeler to describe the motivation underlying it), the identity of the creator, whether it is the baseline scenario for this model version, and the date of creation.

The assumptions associated with running a “what if” scenario are captured by a parameter set, which associates each parameter of the model (whether user-defined or built-in) with a particular value (with a common value for a given parameter being simply the default value for that parameter within the specific model version at hand). Each parameter is also associated with a data type, an optional dimensional product (Szirtes and Rózsa, 1998) specifying the units by which the value is measured, and a motivation for the value specified. In the existing version of the software, parameter sets are always accessed in the context of an associated scenario.

While scenarios within the interface hierarchy are specific to the model version, they may be imported into other model versions, through either a Copy/Cut & Paste or via Drag & Drop metaphors. In the event that the destination model version includes additional parameters beyond those expressed in the parameter set, the default values of those parameters will be used; in the event that that destination model version does not include some of the parameters in the parameter set, those parameters will be dropped. To lower the risk of misunderstanding, the user will be prompted for any such modification of the parameter set when importing the scenario for use with a distinct model version.

A given scenario may be associated with stored simulation results, and the user may elect to perform a simulation at any point. When working with those models created in simulation packages for which automated interfaces are available, the simulation process is transparent to the user. When working with model software packages for which no automated interface exists, SILVER may be used to store away model versions and organize the model project, model version, and scenario hierarchy, but simulations themselves must be performed manually by the user. Specifically, to run a scenario, the user is responsible for instantiating the parameter values, running the model, and directing SILVER to all output files (e.g. dataset files) for storage.

For scenarios that have stored results, the software maintains information on the duration of the simulation (measured in seconds), the time at which the simulation results were entered (whether manually or via automated interface), scenario output files (e.g. datasets for Vensim), as well as (where available) a snapshot of the model user interface at the completion of the run.

Of course, storing model output results does have some cost, with the convenience of ready access being weighed against (amongst other factors) the storage requirements. As a result, the user may choose to delete output files associated with a given previously-run scenario, or to flush all stored information on previous results.

In addition to capturing direct model output for a scenario, the software permits the user to associate a given scenario with zero or more external files. Such files might be input files, or output files (for example, Excel files used to analyze or better visualize scenario results.) Just as for scenario output files, the user may request that such files be recreated on the user’s local computer. Like the other files, the user is then responsible for cleaning up the reconstituted files once complete, but can rest assured that complete copies of the files (as of the time of reconstitution) remain present in the SILVER system.

### 4.2 Collaborative Use

An important goal for SILVER was to facilitate such teamwork by reducing the barriers to distributed understanding and sharing of model artifacts. The software described here can store all modeling information either on a user's local computer (using XML text files), or in a remote repository that is accessible via a web services interface. The use of a remote repository permits multiple users to work on a shared set of modeling artifacts, and provides both parties with the ability to browse and – where appropriate, modify – common model projects, model versions, scenarios and scenario hierarchies. The use of a web service permits navigation of institutional firewalls.

At a concrete level, the provision for such distributed access will permit, for example, multiple parties to browse and comment on a common set of scenarios, and to add scenarios and model versions.

An additional virtue of the ability to support distributed access is the ability of the underlying architecture to be adapted to support browser-based use. The development team hopes to release a web server component supporting browser-based remote use within a year.

### 4.3 Software Implementation

This software was developed at the Department of Computer Science at the University of Saskatchewan. In accordance with the principles of Domain Driven Design (Evans, 2004), the software is structured in an object-oriented fashion (Booch et al., 2007) around domain concepts, particularly those introduced in Section 2. Each such concept is captured through associated software objects, which group and encapsulate related information and behavior. These objects are maintained by the software in a database stored within a single file on disk.

The software described here is currently being built, with alpha-testing anticipated in summer 2008. The software is built using Java version 1.6 with Aspect-Oriented Extensions (via AspectJ (Gradecki and Lesiecki, 2003)), and the Eclipse Rich Client Platform (McAffer and Lemieux, 2006). The use of Java allows for cross-platform support, and offers the potential for enhanced support of remote collaboration via web browser-based use from public computers for which installation of new programs is not allowed. In order to allow for broader community use and contributions, the software will be released for open-source distribution via popular open-source site SourceForge (SourceForge, 2008).

## 5 Discussions & Conclusion

We believe that while the software described here is currently in its infancy, it offers potential for significantly facilitating the System Dynamics modeling process. We anticipate that while some of the resulting benefits are obvious (for example, easier collaboration, greater transparency of parameter spaces explored, reduced clutter), others will be just as important but less immediately noticeable – the errors avoided in scenario definition or interpretation, and added richness in the insights gained regarding the behavior of real-world systems being modeled.

Despite its anticipated open-source availability, the use of the software does not come without costs. The limited APIs provided by most modeling packages means that the software must eke out a somewhat awkward coexistence with simultaneously running modeling software. We originally hoped to design software that would remain in the background while the user ran their chosen modeling package, with our software to automatically and transparently intercepting and saving away model versions, scenario definitions, and scenario output created by the user in the course of modeling. Unfortunately, most simulation software APIs lack the requisite generality. As a result, we currently require modelers to alternate use between their modeling package and SILVER, where the modeling package is used to perform model structural modifications and in-depth investigation of (recreated) scenario results, and SILVER is used to define and run scenarios, to check-in new model versions, and to recreate earlier model versions or scenario results. We hope that if SILVER is successful in attracting modeler attention and support, creators of modeling packages will consider broadening their APIs to better support a richer interaction between SILVER and their software.

The creators of SILVER are acutely aware of the need to expand the functionality of the project to better support additional aspects of the modeling process. For example, the current system does not offer the ability to run Monte-Carlo ensembles or other sensitivity analyses, to define time-dependent parameter settings for scenarios, or to define systematically structured sets of scenarios. The project philosophy was to start simple, in the hopes that a basic set of functionality would facilitate an important subset of the modeling process, with the knowledge that the remainder of the modeling work could still be conducted using traditional tools and conventions, and through manual interfaces with the software.

We invite readers to download the software when it becomes available for beta-testing or once Release 1.0 is available, and to provide feedback or comments (via the website or via email to the lead author) as to the degree to which current functionality matches their needs, and highlighting perceived priorities for added functionality.

## Acknowledgements

The authors wish to thank Ozge Karanfil, Qian Zhang, David Vickers, Vipat Kuruchittham for their helpful feedback on the functionality and visual design of SILVER. Nathaniel Osgood wishes to offer his sincere thanks to the National Science and Engineering Research Council of Canada for the research funding that helped underwrite software development effort for this project.

## Bibliography

Booch, G., R. A. Maksimchuk, et al. 2007. *Object-oriented analysis and design with applications*. Upper Saddle River, NJ, Addison-Wesley.

Evans, E. 2004. *Domain-driven design : tackling complexity in the heart of software*. Boston, Addison-Wesley.

Gradecki, J. and N. Lesiecki. 2003. *Mastering AspectJ:  Aspect-oriented programming in Java*. Indianapolis, Ind., Wiley.

Maplesoft. 2008. Maple. Waterloo, Ontario, Canada, Waterloo Maple Inc.

McAffer, J. and J.-M. Lemieux. 2006. *Eclipse Rich Client Platform : designing, coding, and packaging Java applications*. Upper Saddle River, NJ, Addison-Wesley.

Microsoft Corporation. 2007. Microsoft Excel. Redmond, WA, Microsoft Corporation.

SourceForge. (2008). "SourceForge."   Retrieved June 29, 2008, 2008, from www.sourceforge.net.

Szirtes, T. and P. Rózsa. 1998. *Applied dimensional analysis and modeling*. New York, McGraw Hill.

Ventana Systems, I. 2004. Vensim. Harvard, MA, Ventana Systems, Inc.

Wolfram Research. 2008. Mathematica. Champaign, Illinois, Wolfram Research, Inc.

XJ Technologies. 2007. AnyLogic. St. Petersburg, Russia, XJ Technologies.