Open Source System Dynamics with Simantics and OpenModelica

Teemu Lempinen
VTT Technical Research Centre of Finland
P.O. Box 1000, FI-02044 VTT, Finland
+358 40 571 0126
teemu.lempinen@vtt.fi

Sampsa Ruutu
VTT Technical Research Centre of Finland
P.O. Box 1000, FI-02044 VTT, Finland
+358 40 720 2883
sampsa.ruutu@vtt.fi

Tommi Karhela
VTT Technical Research Centre of Finland
P.O. Box 1000, FI-02044 VTT, Finland
+358 40 582 2274
tommi.karhela@vtt.fi

Peter Ylén
VTT Technical Research Centre of Finland
P.O. Box 1000, FI-02044 VTT, Finland
+358 40 507 7474
peter.ylen@vtt.fi

Abstract – *We will introduce a new system dynamic modelling and simulation environment based on open source components. The development was initiated by a group of active system dynamics modellers who had needs and ideas for an open toolset. The new needs for features like hierarchical modules, module libraries, collaborative model development and efficient model communication in system dynamics together with the development of open source modelling framework Simantics and simulation environment OpenModelica have driven us to start developing an open source modelling and simulation software for system dynamics. In this paper we discuss how current open source components can be used to build a comprehensive tool for system dynamics modelling and what impact open source could have on system dynamics modelling. Even though the development is still on its early stages, the open source components have enabled us to rapidly develop a tool capable of hierarchical modelling, simulation and some basic result and model analysis. When using open source, the modelling software becomes more affordable and distribution of models becomes easier, modelling software can be adapted to individual needs and models can be used and validated by all stakeholders.*

Introduction

Using commercial system dynamics modelling software has been a necessity for professional modellers and there is a wide range of different commercial products on the market. Some of the modelling tools are specialized in system dynamics, while others can even combine different types of modelling paradigms. Even though the commercial products are widely used, we believe that there is demand for a comprehensive open source tool for modelling and simulating system dynamics.

We have been working closely with a team of modellers, who have been successfully using certain commercial system dynamics software for years. The commercial product is working well, but is not responding to all the new needs of the modellers and end users, such as hierarchical models and online user interfaces. Instead of finding a new software, that would again be a closed system and hard to modify, the time was seen right to see if it would be possible to develop an open source modelling tool that would be comprehensive enough to use in real customer projects and easy to develop further.

Open source is not a new subject in the system dynamics modelling world. There are few open source modelling tools available (e.g. Sphinx SD Tools, OpenSim), but they lack advanced functionalities like multidimensional variables and are not intended to be as comprehensive as commercial solutions.

We believe that open source system dynamics modelling tools have been lacking a good framework to build on. Building everything from scratch is an expensive and time-consuming task. The advent of the open source modelling and simulation framework Simantics (Simantics) and the development of open source simulating environment OpenModelica (OpenModelica) have enabled us to rapidly develop a tool that is already capable of simulating large hierarchical models with multidimensional variables and also display these models online.

In addition to the open source components, the field of software development could offer even more to system dynamics. Like our tool, software is usually built from separate and reusable components. The component approach has also been adopted in other modelling and simulation methods (e.g. Modelica and Apros) that mainly focus on physical modelling. Collaborative development, which is widely applied in software industry, could also vastly improve the current system dynamics modelling methods.

There is ongoing research on how reusable components, modules, could be incorporated into system dynamics modelling. The modellers we have been working with often face situations where they would like to use some parts of models they have done in previous projects as basis for new models or replicate model structure in ongoing projects. They have for example been modelling complex and exceptionally detailed models with a lot of repeating structures that have been used to model customers' business processes in product development (Pesonen et al. 2008). This kind of large and complex models could probably benefit from reusable modules and module libraries and thus they are one main focus on our tool.

In this paper, we will discuss the opportunities and challenges related to building open source system dynamics software and reflect them upon our own development effort.

First, the motivation for our software and choosing open source are discussed followed by a short introduction of the functionalities. Then we take a closer look on how the open source community can aid in developing system dynamics software focusing on the two main components: Simantics and OpenModelica. Finally, new concepts are introduced on how the tool and system dynamics modelling could evolve in the future.

Motivation for a new tool

System dynamics has great potential to support decision making in a wide range of fields ranging from environmental and political policy formulation to business process planning. The effective use of system dynamics requires a lot from the modelling software. Our motivation to start developing modelling software arose from the needs of modellers. Some of the concepts described in this chapter have already been implemented in some of the commercial products, but the combination of open source software and this set of features are novel. The goal is to support the whole modelling process from the collaborative modelling process to model validation, documentation and finally communicating the model and the results to the public.

Many system dynamics models are based on general and widely accepted model structures, such as the rework cycle in project management (Lyneis & Ford 2007) and the stock management structure in supply chain management (Sterman 2000), which are modified to fit the individual needs of modellers. In software development, libraries are used for such general components. Similar libraries could also be used in system dynamics modelling - especially with detailed operative models such as project management or supply network models. Module and model libraries would provide tested and validated components to build new models on. Creating large models with great detail could become easier and faster. These libraries will also require an effective distribution and sharing system to mature and be used by many different modellers.

Version control system is an effective way of collaborating in software development. For text-based models, like the .mdl-files produced by Vensim (Vensim), a normal version control system can be used to enable collaboration in modelling system dynamics (Helfrich and Schade 2008). Such a version control system needs to be integrated to the modelling software to support collaboration. The same system could also be used for distributing module and function libraries.

Models are made to be used. Easy distribution and good documentation are important for the use of a model. Models created with open source software are easily distributed, since the software is available for everyone. To make distribution even simpler, the models should be usable through an ordinary web browser in every computer. The complex system dynamic model is not probably the best way to convey the results of a simulation to every audience. Instead, case and user specific user interfaces are needed to show the content that is really important for different users. The documentation of the model should also be as automated as possible and respond to changes.

There are many different mathematical methods that can be applied to optimizing and analyzing a system dynamic model (Oliva 2003, Kaupmann & Oliva 2008). A number of methods need to be built-in to the tool, but also interfaces to external systems are required. External interfaces are essential for data from enterprise resource planning

systems, product configuration systems and other data sources. Open software could enable integrating any software when needed.

Why open source?

Open source system dynamics software has advantages compared to more traditional commercial software. First of all, open source software is free for the user. By providing software free, more students and practitioners can get acquainted with system dynamics. When more people have experience of system dynamics, it is more easily used also in important real-life decision making.

System dynamics modelling is best done in a reflective way, “in which testing is designed to uncover flaws and hidden assumption, challenge preconceptions, and expose assumptions for critique and improvement” (Sterman 2000, p.858). Publishing model files is useful, and free software such as Vensim Reader (Vensim Reader) and iSee Player (iSee Player) are designed for people to run simulation models. There are also free system dynamics simulators available, such as Vensim PLE, that allow users to modify models, but their use is restricted to simple models without such features as array variables and hierarchical models. We claim that the use of open source simulation software is one step forward towards the desired reflective mode of modelling. Open source software not only allows a wide community of people to run the model and vary individual parameters, but also lets people to modify the structure and boundary of the model.

The price of the software is a factor when new users try out system dynamics. With professionals, the costs of the actual modelling process greatly overrun the cost of the software. Even in large and expensive projects, there could still be need for open source software. The customer can be more easily integrated to the modelling process, if they also have some means to explore and modify the model. Purchasing software licenses for many users for a single project may not be in their interests. With open source software, purchasing the software and integrating customers to the process will not be a problem. This boils down to a fact that the nature of modelling and simulation is more service business than license business. Thus it is natural to use open source tools for it.

Open source software also enables the software to be modified for the individual needs of the users. If a new feature is needed, it can be produced by anyone. New analysis tools or interfaces to company-specific systems, such as Enterprise resource planning systems, can be produced faster.

There are, however, a few drawbacks with small and young open source systems. The user support relies heavily on other users and the developers of the software. At the beginning, the community is small. The more users the system gets, the easier it is to get help from others. Also, the quality of the software may suffer from a small audience. Since money is not earned by creating the software, testing and other quality efforts may not be at the level of commercial products. Again, a larger community of users can provide more resources to quality assurance and testing of the software.

The software

Our tool for system dynamics modelling is currently under development. It uses Simantics as the user interface and database, Modelica as expression language and OpenModelica for compiling and simulating models. Even though the software is young, it is capable of modelling and simulating large hierarchical models with multidimensional variables. The model structure and simulation results can be analyzed with visual tools and basic operating interfaces can be created for using the models.

The models are created in a traditional way using a graphical user interface with stock & flow diagrams and causal loop diagrams. Modules and interface variables have been added to the basic modelling syntax to support hierarchical modelling (Fig 1). There has been no need to radically change the syntax that modellers have already used to work with.

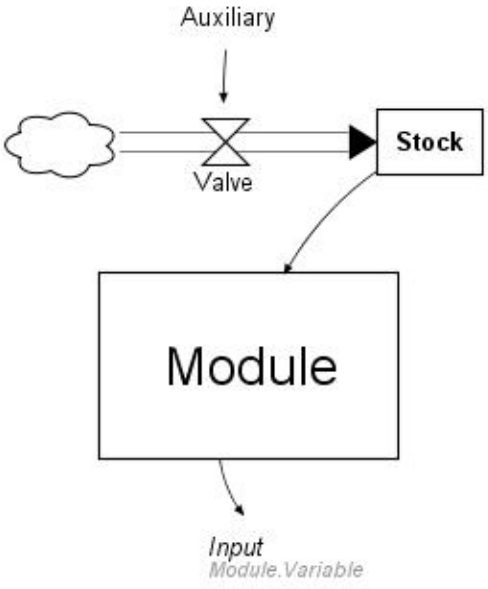


Fig 1. Modelling syntax

The purpose of hierarchical modules is not just to encapsulate and hide some functionality. Many module structures are general and can also be reused in different models. They also provide a good starting point for more specialized modules. Module libraries provide a convenient way for storing and sharing these modules. Reusing and extending components is a key element in object-oriented programming. It enables fast development using robust and tested components.

The models are stored in Simantics' semantic database instead of more traditional text files. The semantic structure contains a lot of information about the model. Based on this information, a textual representation in Modelica language is created for simulations. The text file is compiled and simulated with OpenModelica. The results are then read back to the modelling software for analysis.

The user needs very little knowledge about Modelica and never needs to see the generated Modelica code. The mathematical syntax used for defining equations is

universal and the equations are defined in an equation editor (Fig 2). The built-in functions in Modelica may need some getting used to, but most of those are common in all mathematical languages. Knowledge about Modelica helps when creating more complex equations and structures especially with multidimensional variables.

Fig 2. Variable properties and equation editor

Open source components

The open source community has created many components that can be used without having to implement everything from scratch. The two main components of the modelling tool are the Simantics platform and the OpenModelica environment. Simantics platform serves as the database and the user interface framework whereas OpenModelica is used for simulating the models.

Simantics

Simantics is an open source modelling and simulation platform, which is currently under development. By using this modelling platform, the system dynamics tool can take advantage of the features common to most modelling environments. The basic user interface components, such as the model structure tree and the diagram editor can be easily modified to fit the needs of system dynamics modelling. Simantics platform provides a semantic database, an Eclipse framework (Eclipse) based client software and other data handling functionalities.

The use of the semantic database and common semantic structures enables the system dynamics tool to use the features developed for other modelling environments, since the data is structured the same way in all contexts. The semantic database does not show directly to the user, but enables developers to use features initially created for other applications, a user-friendly and efficient way to browse content and relations in the database and easy integration to other systems.

The purpose of Simantics is to support all types of modelling and simulation needs as an integrating platform. It does not contain any equation solvers itself, but the framework has been created to support integration of any solver tool. Dynamic simulator (Apros), steady state chemical process simulator (Balas) and a discrete event simulator are examples of different simulation types for which tools are being developed on top of the Simantics platform. Many of the products are not open source, even if the framework is. They still contribute components to the overall framework developing it further.

OpenModelica

We are using the open source OpenModelica environment to simulate the system dynamics models. The models are written in Modelica language and OpenModelica is used to compile and simulate the Modelica code.

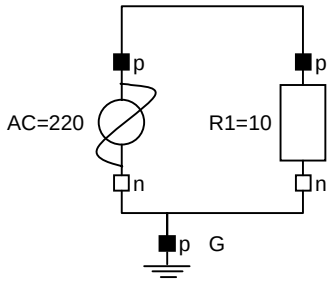
Modelica language

Modelica is an object-oriented, equation based modelling language that has originally been created for modelling and simulating physical systems. Modelica syntax is similar to other programming languages like Java.

A simple electrical circuit model (Table 1) serves as an example of the use of Modelica language. With Modelica editors, a circuit can be defined visually using electrical component libraries.

Table 1. Electrical circuit model (adapted from Fritzson 2003)

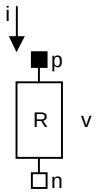
Visual Modelica model	Modelica code

	`class Circuit`
	`  Resistor R1(R=10);  //Resistance=10`
	`  VsourceAC AC(VA=220); //Amplitude=220`
	`  Ground G;`
	`equation`
	`  connect(AC.p, R1.p);`
	`  connect(R1.n, AC.n);`
	`  connect(AC.n, G.p);`
	`end Circuit;`

Each component is defined as a separate class. An example component Resistor is shown in Table 2.

Table 2. Resistor component in Modelica library

Visual Modelica model	Modelica code

	`class Resistor`
	`  Pin  p, n;`
	`  Voltage  v;`
	`  Current  i;`
	`  parameter Real R(unit="Ohm");`
	`equation`
	`  v = r * i;`
	`  0 = p.i + n.i;`
	`  i = p.i;`
	`end Resistor;`

Modelica is well-suited for this kind of modelling. Using the Modelica language for system dynamics is also quite natural. System dynamics models are basically just differential equations that can easily be represented in the Modelica language. Since the structure of the model is stored in the database, there is no need for each variable to be an instance of a special class and connections to be made from one Pin to another.

In basic models, the mapping from the model structure in the database to Modelica language is quite straightforward. All variables in the system dynamics model are usually declared as real numbers. The equations of the variables are written in Modelica language and used directly in the equations-section of the Modelica-model. The mapping from system dynamics models to Modelica is explained in the following section.

There are other projects based on the Simantics platform that are also using the OpenModelica environment. Collaborating with those projects enables us all to use the same components for simulation and reuse our work. The same reusability goal we seek to achieve also in the system dynamics modelling. In addition to using the OpenModelica components in many simulation environments, the Simantics platform may enable the integration of other types of models with system dynamics.

Mapping system dynamics to Modelica

The next sections are focused on the technology under the hood of our system dynamics tool: how system dynamics models are mapped to Modelica and how Modelica can be used to enhance system dynamics modelling. In many of the example cases the new simulator is compared to Vensim just for the sake of reference. Also, the models presented in the examples are generic archetype model structures used by many authors (e.g. Stermann 2000) in different contexts - the population model in fig. 4 and the work model in fig. 8 to name but a few.

System dynamics library for Modelica

A system dynamics library has already been created for Modelica (Modelica System Dynamics). The library can be used by any Modelica compliant Integrated Development Environment (IDE). However, the library is restricted with the physical nature of the traditional models created in Modelica. Every connection needs a dedicated port in a variable and thus modelling with the library becomes overly complicated. Also, the visual representation differs from the more classical stock and flow diagrams, which are very efficient for communicating feedback loops. Our solution is to separate the visual model configuration and the actual simulation code to achieve better user experience. System dynamics modellers can work with familiar diagrams and do not have to worry about the restrictions of using Modelica for both the visual presentation and the simulation code. They do not have to worry about the Modelica code at all, since it is generated automatically based on the models.

Modules

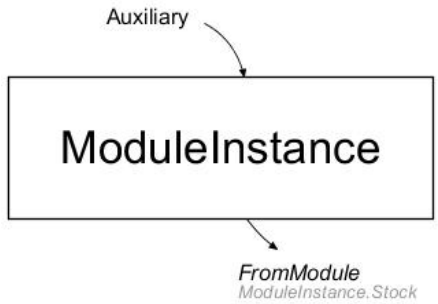
The object-oriented nature of the Modelica language provides a natural solution for the implementation of hierarchical modules. Each module type is defined as a class. Module instances are instances of those classes. In this section we will discuss how the Modelica language relates to some concepts in system dynamics modelling.

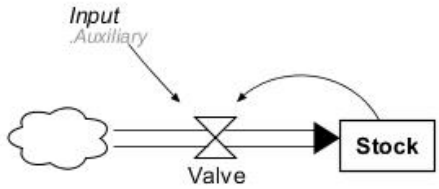
Adopting this type of object oriented approach to system dynamics modelling can significantly speed up the development of large and complex models. The modules clearly and logically separate different functionalities in the model. Reusing the created module is simply a matter of dragging another instance of the module to the diagram and defining the input and output variables.

Below is an example of a small model with a simple module (Table 3). The Modelica representation of the model is generated automatically when the model is simulated. Users do not have to be familiar with (object oriented) programming. The equations for variables are defined in the properties of each individual variable and automatically added to the Modelica representation.

Table 3. An example hierarchical model and automatically generated Modelica code.

System dynamics model	Automatically generated Modelica code

	`model Model`
	`// Variable definitions`
	`  parameter Real Auxiliary = 5;`
	`  Real FromModule;`
	`// Module definitions`
	`  ModuleType1 ModuleInstance;`
	`equation`
	`// Inputs and outputs`
	`  FromModule = ModuleInstance.Stock;`
	`  ModuleInstance.Input = Auxiliary;`
	`end Model;`

	`class ModuleType1`
	`// Variable definitions`
	`  Real Valve;`
	`  Real Stock(start=1.0,fixed=true);`
	`// Input definitions`
	`  Real Input; // Value defined in Model`
	`equation`
	`// Equations`
	`  Valve = Stock / Input;`
	`  der(Stock) =`
	`  + Valve;`
	`end ModuleType1;`

In a small model like this, it would probably be useful to display all the variables in a single diagram. The modules can, however, hide even very large structures. Reusing those large structures is just as simple as reusing the simple structures. Only the interface variables need to be defined. When changes are made to the module type, all the changes are effective also in the instances of the modules.

Functions

The benefits of using Modelica go beyond the use of classes to represent module types. Similar to module libraries, function libraries can also be created easily. Creating functions require more knowledge about Modelica than the basic graphical modelling, but the benefits are similar as in reusable modules.

Sometimes complex equations are written over and over again for different variables. By creating a function that implements the same equation, the function can be used

instead of the whole equation. In addition to saving time typing or copying the equations, functions can be given representative names that explain the behaviour better than the complex equation.

An example of such function could be a function frequently used by Vensim modellers: ZIDZ (zero if divided by zero). ZIDZ provides a convenient way to avoid dividing by zero.

Modelica even enables the use of external functions written in C programming language. External functions are especially useful for complex functions that can be created in mathematical software or already exist in public libraries.

Multidimensional variables

One very popular way of replicating model structure is to use array variables. Modelica supports array types, so implementing multidimensional variables is not difficult. Modelica even includes many built-in functions that can be used to handle the multidimensional variables.

Multidimensional variables are a challenge when using modules. The creator of the module may want to support multidimensional variables, but does not necessary know how large variables the user of the module wants to have. Since modules can be used in different models, a global enumeration for indexing cannot be used. Instead, enumerations are defined separately for each module.

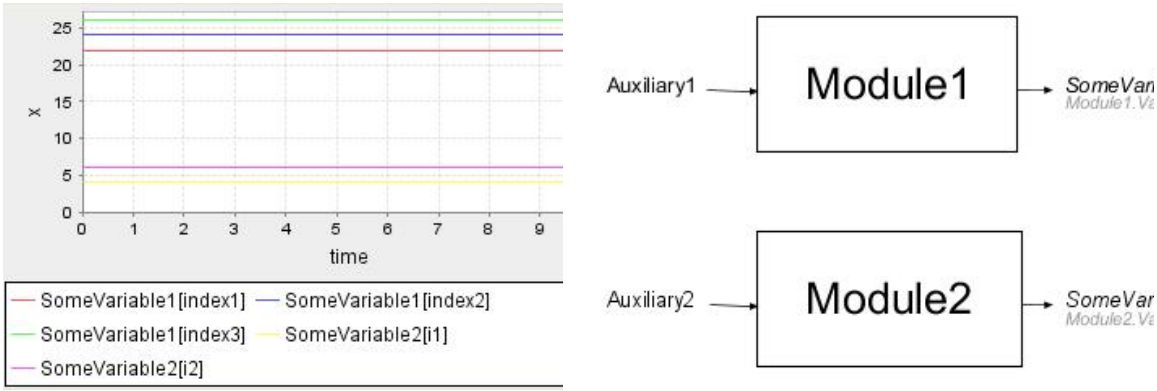


Fig 3. Different dimensionalities in different module instances

Users can override the enumerations for each instance of a module. This way the modeller can have multiple instances of the same module, but inside those modules are array variables with different dimensionalities. It is the responsibility of the module creator that equations inside the module are applicable for array variables of different sizes. Individual array indexes cannot be used in equations. An example of two module instances with different dimensionalities is presented in Fig 3.

In the upper chain, an enumeration with three indexes is used.

```
Auxiliary1[3] = {10,11,12};
```

In the lower chain, variables are assigned an enumeration with two indexes.

```
Auxiliary2[2] = {1,2};
```

The original enumeration in the modules is replaced separately for each instance. There can be multiple different enumerations used in a module and they can all be replaced depending on the case where the module is used. This enables great reusability of the module.

Validating simulation results

To make sure that OpenModelica will produce correct results for system dynamic simulations, a number of validation tests have been made. An example test is the basic population model for growth & overshoot (Sterman 2000, p.121). The model consists of a growing Population, Carrying capacity and a delay in the feedback loop that indicates if the capacity has been reached (Fig. 4).

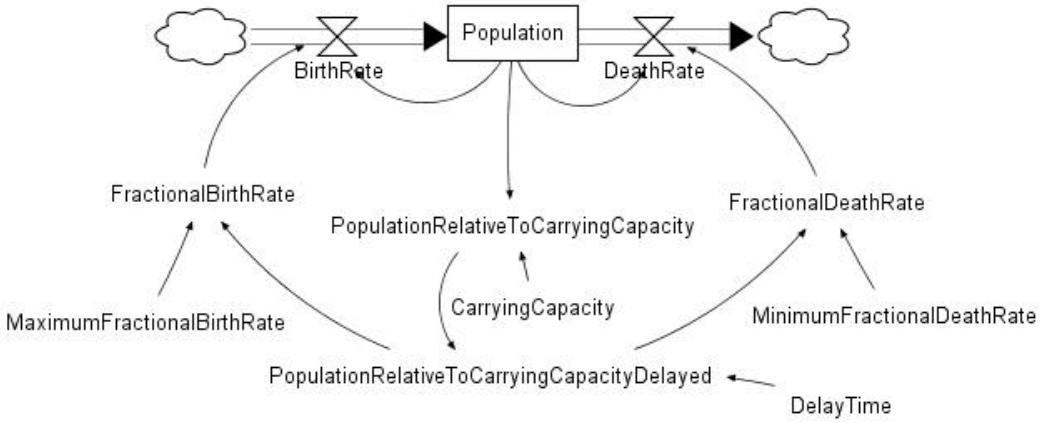


Fig 4. Growth & overshoot model (adapted from Sterman 2000)

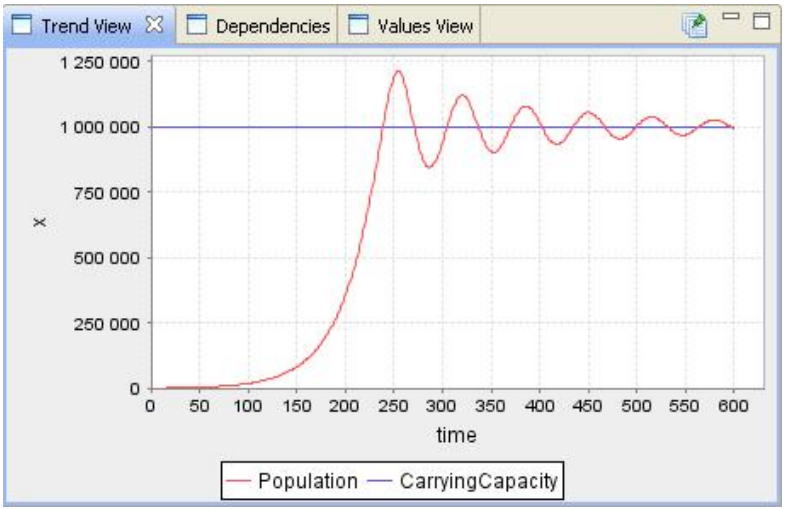


Fig 5. Results of the simulation of growth & overshoot model

The results of the simulation show the oscillation caused by the delay (Fig 5). When the same model is simulated in Vensim, the results are a close, but not exact, match due to different solvers (Fig 6). Models with delays and complex structures usually produce similar results with Vensim and OpenModelica.

OpenModelica provides the possibility to use different types of solvers. Included solvers are DASSL, Euler and Runge Kutta 4. This comparison was conducted using Euler in both simulators. Euler is not the best method to use in this type of oscillating models and using it may cause unwanted behaviour in the model. However, our experience is that Euler is used as a default solver in spite of its characteristics and thus we are using it also in our comparison. It has not been the scope of our research and work to evaluate the accuracies of different solvers, but even with large models, the results are close enough to say that OpenModelica can be used to simulate system dynamic models.

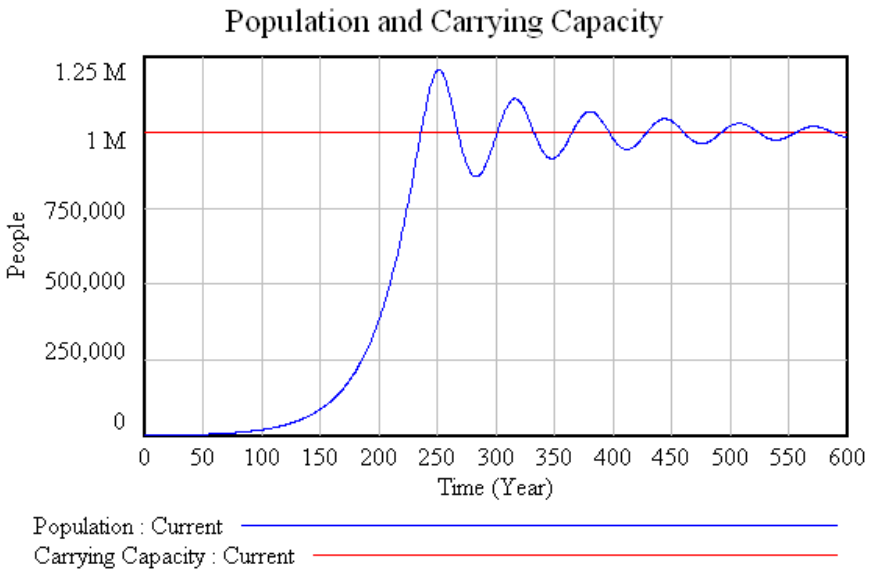


Fig 6. Results of the simulation of growth & overshoot model in Vensim

Other components

Simantics and OpenModelica are the largest and most important components for the functionality of the system dynamics tool. Other open source components have also been used for smaller tasks. Visualization of simulation results has been implemented using JFreeChart library (JFreeChart). Currently only time series are supported (Fig 7), but JFreeChart consists of a variety of different highly modifiable charts. Other diagram types will be supported in the future, but since the tool is open source, users can even create their own charts for special needs.

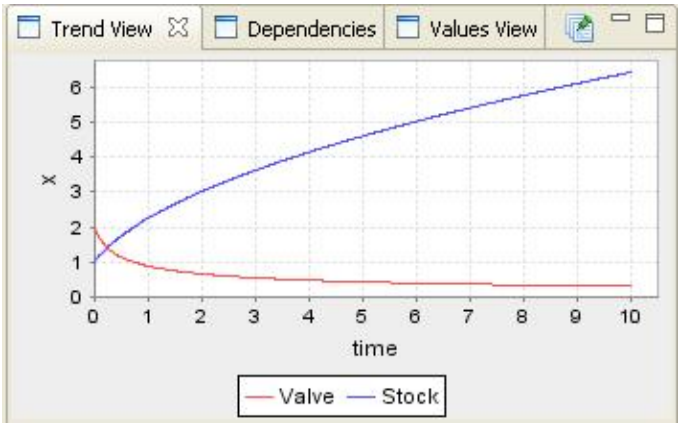


Fig 7. Result visualization using JFreeChart

Graphviz (Graphviz) is an open source graph visualization software that is used to display dependencies between variables. Like JFreeChart, Graphviz is a widely used and highly matured component. Graphviz is especially suitable for displaying structural information (Fig 8). For other visualization needs, the world is full of different tools that can easily be integrated to the modelling software.

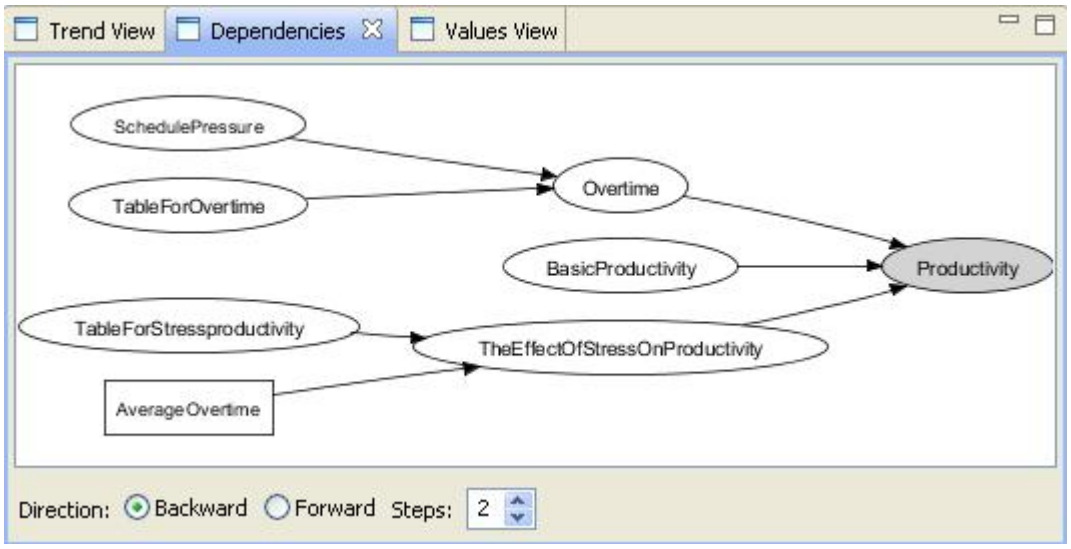


Fig 8. Dependencies visualization using Graphviz.
(Model adapted from Vensim Modeling Guide: Project dynamics, distributed with Vensim software)

Future development

Using an open source modelling platform has advantages in the early stages of product development, but also in the future. New functionalities are developed all the time for different modelling needs and in different projects. Many of those are created to be general tools for all modelling environments. In this chapter we will take a look at some of the things to come.

Visualization of model structure and simulation results

System dynamics models contain vital information about the structure of the systems and their behaviour over time. For communicating this information, system dynamics

tools need extensive visualization options for both model structure and simulation results.

Currently our tool contains only the basic model editor, a dependency view for structure visualization and table and time series views for result visualization. So far the development has focused on creating models that can be simulated. A master's thesis project has recently been started that purely concentrates on visualizing the simulation results in various ways. The outcomes of the project include visualization options and profiles for the diagram editor and also different types of modifiable charts for visualizing simulation results.

Mathematical analysis

Simantics aims at providing an efficient integration platform for simulation and analysis tools. Tools like sensitivity analysis and optimization have been planned for this system dynamics tool. OpenModelica compiles the model to an executable, which has easily modifiable parameters, but the model itself is simulated quickly enabling different types of analysis methods. The same mathematical analysis tools can be applied to all applications using Simantics and OpenModelica.

Team features

Software has long been developed in groups using version control systems. Version control systems enable distributed development and serve as a reliable centralized backup. Traditional version control systems operate with text files, so they are not applicable for the semantic database in Simantics. There are plans to create similar team features also for the Simantics platform. Once they are implemented, the system dynamics tool can also take full advantage of these features.

In addition to managing your own projects, these repositories of models can also serve as a distribution channel for the function and module libraries. Libraries can directly be synchronized with the modelling software and updates to the libraries are easy to make.

Online models and user interfaces

System dynamics models can be used to educate people and convey information. What better way to distribute information than the internet? A commercial product called Simupedia is being developed for distributing models created with products based on the Simantics platform. There is functionality in the Simantics platform that enables users to create user interfaces for their models. The user interfaces can be used with the modelling products or published in the Simupedia communities.

Connectivity

In all kinds of modelling, there is a need to use accurate and up-to-date data in simulations. The data can be found and created in various databases, files, sensors and software. In addition to the more traditional connections such as an interface to excel sheets, there is also an on-going research project on how to combine data from a product configurator to system dynamics modelling tool. The aim of the semantic database in Simantics is to provide a way to represent data related to a model semantically and independent of simulators. Many external connections are created for other purposes

than system dynamics modelling, but the use of semantic data representation enables the system dynamics tool to use the same interfaces as well.

Model documentation and spreadsheets

Documenting models and results is crucial for all simulation. A wiki documentation tool is currently being developed for Simantics that would enable documenting model structure and creating templates for simulation reports. The documentation tool will contain blocks that are automatically updated when the simulation results, graphical representation or some parameters are changed.

Spreadsheets are a convenient way of receiving data from simulations but also providing model parameters. A spreadsheet tool is also being created to the Simantics platform. With the spreadsheet tool, model parameters can easily be configured in a single place. The spreadsheets can be embedded to editors, model diagrams and other views in the Simantics environment.

Discussion

Developing open source modelling tool based on other open source components has been relatively fast and rewarding. Using Simantics and the Eclipse framework has provided a lot of opportunities in the user interface. Many components are already provided by the frameworks and also the user interface style and interaction guidelines are set. What has been left to do is to adapt these to correspond to the needs of system dynamics modellers.

Using OpenModelica provided an easy way to simulate system dynamic models. Instead of building a completely new simulation engine, only a mapping from system dynamic models to a representation in Modelica language was needed. The object-oriented nature of Modelica language has supported the needs for both reusable modules and multidimensional variables.

Even though reusable modules can be created, they should not be seen as solutions to all possible situations. System dynamics is about understanding the problem by modelling it. The problem should not be forced to fit any ready-made moulds. However, many situations are similar to each other. Modules in module libraries could be seen as suggestions on how to handle certain types of modelling problems. If they do not describe the problem completely, they should be extended or modified. The libraries could even contain different versions of the same module to better describe various situations. Modules may also be intended to be reused only in one model, when the modeller realizes that the model will contain repeating structures.

Both Simantics and OpenModelica are still under development and subjects to change. This has lead to some challenges in the development. Simantics platform is young, so even large parts of the platform can change and they need to be updated. On the other hand, since the platform is being developed all the time, the outcome can be influenced. The system dynamics tool has influenced a lot of the features, since previously the platform has concentrated on modelling physical models. System dynamics models are more abstract and for example connections do not necessarily have properties and they do not need dedicated ports.

The current version of OpenModelica supports almost everything essential for system dynamics, even though there have been some issues with for example using enumerations as array indexes. This is very lucky for us, since we do not develop OpenModelica and the development is mainly driven by different kinds of simulation needs. One big issue in development has been the solvers in OpenModelica. It is not that they wouldn't work, but the solvers are very sensitive about discontinuities. The group of modellers that has participated in the development of our system dynamics tool has worked mainly with Vensim. The default solver in Vensim handles discontinuities and other difficult situations differently than OpenModelica. Some models that we have used for validating the functionality of our software have contained situations that OpenModelica just cannot handle. It is not that hard to change the models, changing the way you model is more difficult.

The main development has focused around Simantics and OpenModelica, but as discussed earlier, other components have also been used. Components like JFreeChart and Graphviz are mature and have been easy to integrate to the software.

An important thing to point out with these components and the whole software is that the system dynamics tool can be developed further by programmers that are not professionals or experts. Quality components and the support of the community can enable almost anyone with some programming skills to create new features.

Conclusion

There are many factors that favour the use of open source software for modelling system dynamics. Models created with open source software can more easily be distributed, validated and criticized, and customers are more easily integrated to the modelling process. Modellers can participate in the development of their own tool and new features are easily created.

The development of open source software depends on an active community and quality components. Previously there has not been a platform that would provide such a good starting point for the development of an open source system dynamics modelling tool. Simantics and OpenModelica have enabled a rapid development of modelling software that is capable of basic graphical modelling and simulation. In the future the software will evolve with new features of the Simantics platform and also with new open source components that can be used for example in visualizing simulation results.

References

Apros, www.apros.fi, accessed March 15, 2011

Balas, http://balas.vtt.fi, accessed March 15, 2011

Eclipse, www.eclipse.org/, accessed March 15, 2011

Fritzson, P. 2003. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Computer Society Pr.

Graphviz, www.graphviz.org, accessed March 15, 2011

Helfrich, N. and Schade, W. 2008. *Bringing distributed software development to SD modelling with Vensim*. International Conference of the System Dynamics Society. Online conference proceedings : July 20-24, 2008, Athens, Greece

iSee Player, http://www.iseesystems.com/softwares/player/iseeplayer.aspx, accessed March 15, 2011

JFreeChart, www.jfree.org/jfreechart, accessed March 15, 2011

Kampmann, C. E., and R. Oliva. 2008. Structural Dominance Analysis and Theory Building in System Dynamics. *Systems Research and Behavioral Science* 25, no. 4: 505-519.

Lyneis, J. M., and D. N. Ford. 2007. System dynamics applied to project management: a survey, assessment, and directions for future research. *System Dynamics Review* 23, no. 2-3: 157-189.

Modelica System Dynamics, www.modelica.org/libraries/SystemDynamics, accessed March 15, 2011

Oliva, R. 2003. Model calibration as a testing strategy for system dynamics models. *European Journal Of Operational Research* 151, no. 151: 552-568.

OpenModelica, www.openmodelica.org, accessed March 15, 2011

OpenSim, www.opensimproject.org, accessed May 25, 2011

Pesonen, L. Salminen, S. Ylén, J-P. Riihimäki, P. 2008. *Dynamic simulation of product process*. Simulation Modelling Practice and Theory. Volume 16, Issue 8, EUROSIM 2007, September 2008, Pages 1091-1102

Simantics, www.simantics.org, accessed March 15, 2011

Sphinx SD Tools, www.sphinxes.org, accessed May 23, 2011

Sterman, J. D. 2000. Business dynamics: Systems thinking and modeling for a complex world. Boston: Irwin McGraw-Hill.

Vensim, www.vensim.com, accessed March 15, 2011

Vensim Reader, www.vensim.com/reader.html, accessed March 15, 2011