SDLPS, a SDL distributed simulator.

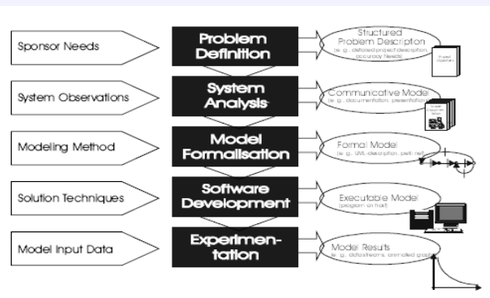
Pau Fonseca i Casas (pau@fib.upc.edu - http://www-eio.upc.es/~pau/)

Abstract

Formalisms allows the complete understanding of a simulation model and helps in its implementation. However only few simulation tools allows an automatic construction of a simulation model based in a formalization of the system. SDL is a modern object oriented formalism that allows the definition of distributed systems. It has focused on the modeling of reactive, state/event driven systems, and has been standardized by the International Telecommunications Union (ITU) in the Z.100 recommendation Since it is a graphical formalism simplifies the understanding of the model. In this poster we show an implementation of a simulation infrastructure that follows SDL formalization language. This infrastructure allows a distributed simulation of the models without any modification to the model definition. Since this infrastructure follows the SDL language formalism it is useful not only for a production use, but to teach formalisms and distributed simulation concepts.

1. Formalization of a simulation model

The construction of a simulation model sometimes lacks in the formalization process needed to understand the model before any implementation. This understanding of the model behavior helps in the implementation process and in the communication between the different personnel involved in the model construction. Also can be considered a product itself (Brade D. 2000). Different formalisms exists in order to represent a simulation model, like Petri Nets or DEVS among others. Some tools have been build in order to allows help the model implementation from the specification (Praehofer and Pree 1993, De Lara and Vangheluwe 2002) and some allows the distribute execution of the models like CD++ (Wainer and Chen. 2003). The proposed infrastructure allows the implementation of a simulation model following the SDL language. Since SDL language allows the definition of distribute systems the resulting model can be executed over different computers without any modification of the model specification.
The infrastructure is implemented in C++.



The formalization of the model can be considered as a product itself (Brade D. 2000).

2. Why SDL

SDL is a powerful and modern language widely used in different scopes, not only in simulation area. It has been standardized by the International Telecommunications Union (ITU) in the Z.100, and can be used easily with UML.
Anyway is not our purpose to argue about what is the best formalism to be used to represent a simulation model. We think that all the formalization languages are good if are useful to simplify and understand the modeling process. Also, exists methods to transform from DEVS to SDL formalism (Fonseca and Casanovas 2005) and vice versa (Fonseca 2006), allowing the use of this infrastructure to models that use DEVS to represent its behavior.

3. SDL formalism

SDL is an acronym of Specification and Description Language. Is an object-oriented, formal language defined by the International Telecommunication Union – Telecommunication Standardization Sector (ITU–T) (formerly *Comité Consultatif International Télégraphique et Téléphonique* [CCITT]) as Recommendation Z.100 (ITU-T Z.100 1999). The language is designed to specify complex, event-driven, real-time, interactive applications involving many concurrent activities using discrete signals to enable communication (Reed 2000, Sanders 2000, IEC).
The definition of the model is based on different components:

1. Structure: system, blocks, processes and processes hierarchy.
2. Communication: signals, with the parameters and channels that the signals use to travel.
3. Behavior: defined through the different processes.
4. Data: based on Abstract Data Types (ADT).
5. Inheritances: to describe the relationships between, and specialization of, the model elements.

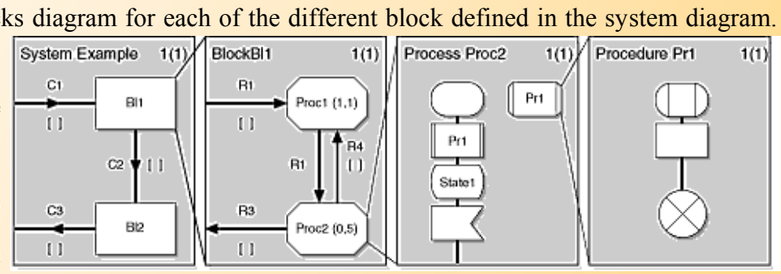
The language has 4 levels (i) System, (ii) Blocks, (iii) Processes and (iv) Procedures

SDL system diagrams

System diagrams represent all of the objects that make up a model and the communication channels between them. A system is the outermost agent that communicates with the environment

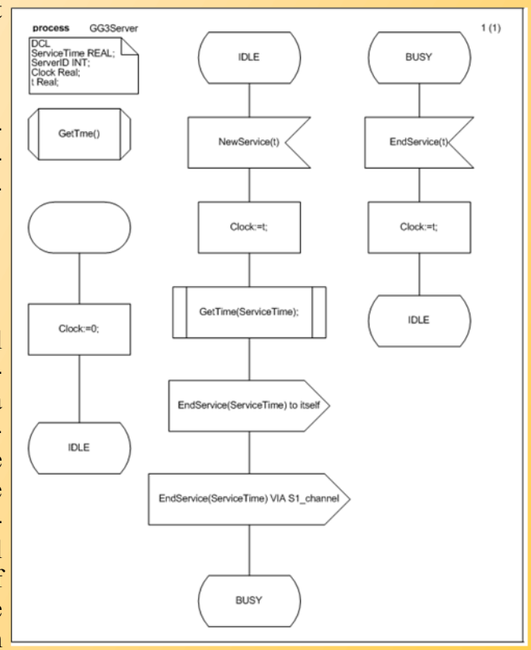
SDL Blocks diagrams

The next stage in SDL specification is the construction of a blocks diagram for each of the different block defined in the system diagram. Each rectangle represents an object. The lines that join the objects are the communication channels (bidirectional or unidirectional communication elements). The channels are joined to the objects through ports. Ports are very important elements for implementing and reusing objects, since they ensure the independence of the different objects. An object only knows its own ports, which are the doors through which it communicates with its environment. An object only knows that it sends and receives events using a specific port. Each block has a name specified by BLOCK keyword. The blocks diagram contains a number of Processes and may also possibly contain other BLOCKs (but not mixed with Processes). Processes communicate via Signal Routes, which connect to other Processes or to Channels external to the Bloc



SDL processes

The processes describe more specifically the behavior of the block. Each one of the processes of the block has one or more states. For each one of the states of a process, SDL describe how it behaves if different events occur. An object may react differently to an event depending on the port that sends it. The process is basically specified using graphical elements that describe operations or decisions.

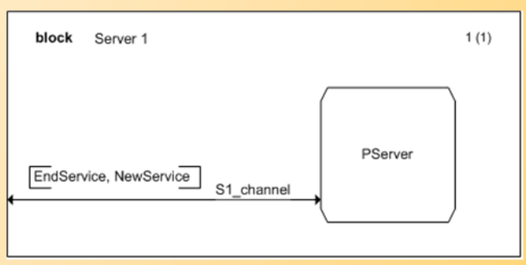


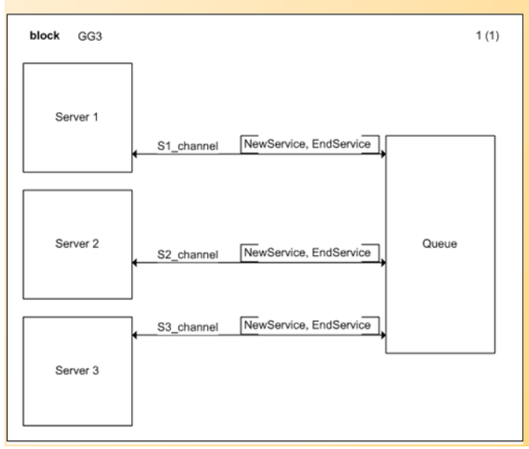
SDL procedures

The last level of the SDL method is the description of the different procedures that appear in the SDL diagrams. These diagrams help describe and specify the model by detailing its most important aspects at the needed level, depending on the target of the specification requirements.

4. SDLPS model formalization

In this section we present a formalization of a simple GG3 model, following the Kendall notation (general distribution for the arrivals and for the services times, and three servers). In our approach the signals that are send from one element to other always have a parameter, with the time representing when this event must be executed. Other parameters can be defined, like priority. In this example however no priorities are defined, hence only the time parameter is needed. The representation of the SDL formalization of the model is done through XML. Although a no graphical version of the SDL language exists, (SDL/GR is the abbreviation for the graphical SDL and SDL/PR for the textual SDL), the use of XML simplifies the management of the language structures and its transformation and manipulation in the infrastructure . Implementation tag in XML file allows the reuse of legacy simulation elements.



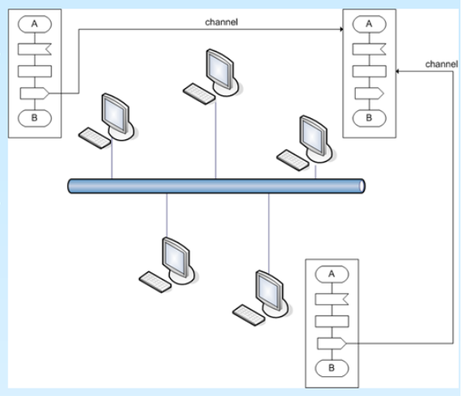


XML representation of GG3 block

```
<block name="GG3" implementation="">
  <channels>
        (..)
 </channels>
 <block name="Queue" implementation="">
        (..)
 </block>
 <block name="Server1" implementation="">
        (..)
 </block>
 <block name="Server2" implementation="">
        (..)
 </block>
 <block name="Server3" implementation="">
        (..)
 </block>
</block>
```

5. System architecture

The system, implemented in C++, allows the distribute execution of different SDL blocs or processes in different machines. The communication is defined through the blocks channels. Each one of the different blocs implements a port and a set of input and output events that can be used to communicate with the other model blocs. Inside each block others blocks or processes can be defined, following SDL formalism.



Time management

In our system a conservative approach for a distribute simulation model has been implemented. Each one of the different channels that connect the elements of the model implements an event list. This method can be reviewed in (Fujimoto 2001).

The compiler

Since the SDL diagrams allows the use of TASK blocs or PROCEDURE blocks, the system must use a C++ compiler in order to allow the execution of the code that the user adds in the specification. We are using MinGw (http://www.mingw.org/) compiler to generate a DLL containing the code related to the specification.

6. Conclusions and future work

This poster presents an infrastructure based in SDL language. This infrastructure allows a distributed simulation of the different elements defined in the SDL formalism. Also an specification of a GG3 simulation model using SDL language is presented. A XML representation of this model is shown. This representation allow the easy use of the SDL model in the system. Also the XML allows the definition of where is the implementation of some elements, allowing the use of legacy simulation models or specific implementation that not follow this infrastructure. Since the infrastructure manages the time and the resources needed to execute the simulation, the user only must describe the behavior of the model without the need of thing if the execution will be local or over different computers. Once the model is constructed each one of the different SDL elements can be executed in a different machine with no more development cost. Also, this infrastructure is very useful to teach the principles of distributed simulation and formalization.

7. References

⇒ Brade D. 2000, Enhancing modeling and simulation accreditation by structuring verification and validation results, Proceedings of the 2000 Winter Simulation Conference J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

⇒ De Lara, J., Vangheluwe, H. 2002, ATOM³: A Tool for Multi-formalism Modelling and Meta-modelling, *Procediings of the 4th International Conference on Enterprise Information Systems ICEIS 2002*

⇒ Fonseca P., Casanovas J. 2005. Using SDL diagrams in a DEVS specification, *The Fifth IASTED International conference on Modeling Simulation and Optimization*, MSO 2005.

⇒ Fonseca i Casas, Pau; Casanovas, Josep; Montero, Jordi. 2004 Adaptación de modelos de simulación estándar a modelos virtuales y/o sistemas de entrenamiento distribuidos, con representación del movimiento continúo de entidades *Revista Iberoamericana de Sistemas, Cibernética e Informática. Num 2. Vol 2*. Available via http://www.iiisci.org/Journal/riSCI/Abstracts.asp? [accessed April 2008]

⇒ Fonseca, P. 2006, Transforming SDL diagrams in a DEVS specificaction. Proceedings of Modelling, Simulation, and Optimization - MSO 2006

⇒ Fujimoto, R. M., 2001, Parallel simulation: parallel and distributed simulation systems, *Proceedings of the 33nd conference on Winter simulation*, Pages: 147 - 157, ISBN:0-7803-7309-X

⇒ ITU-T Z.100. 1999 International Telecommunication Union, Telecommunication standardization sector of ITU, Specification and Description Language (SDL), Series Z: Languages and general software aspects for telecommunication systems. Available via http://www.itu.int/ITU-T/studygroups/com17/languages/index.html [Accessed April 2008].

⇒ Praehofer, H. Pree, D. 1993. Visual modeling of DEVS-based multiformalism systems based on higraphs. *In Proceedings of the 25th Conference on Winter Simulation* (Los Angeles, California, United States, December 12 - 15, 1993). G. W. Evans, M. Mollaghasemi, E. C. Russell, and W. E. Biles, Eds. WSC '93. ACM, New York, NY, 595-603. DOI= http://doi.acm.org/10.1145/256563.256737

⇒ Reed, R. 2000. SDL-2000 form New Millenium Systems, *Telektronikk* 4.2000 p. 20-35

⇒ Sanders, R. 2000 Implementing from SDL, *Telektronikk* 4.2000 p 120-129

⇒ Wainer G., Chen. 2003 W.A framework for remote execution and visualization of Cell-DEVS models. Simulation: Transactions of the Society for Modeling and Simulation International. November 2003. pp. 626-647.