Development of an advanced JAVA based Simulation Tool

Wolfgang Kühn
Susanne Ose
University of Wuppertal
Rainer-Gruenter-Str. 21
D-42119 Wuppertal, Germany
E-mail: wkuehn@sipoc.de

KEYWORDS

Simulation library, discrete event simulation, object oriented simulation

ABSTRACT

On top of the object oriented discrete event simulation library JAVA-Sim, developed at the University of Wuppertal, a modern comfortable graphical user interface GUI has been designed. For the ease of use the modeling interface has a flexible and easy to use control concept. The user interface is strictly separated from the core simulation classes. In order to feed the demand of a general simulator as well as the demands of an easy to use print-media-production simulator a consequent design and class structure has been required. Therefore the priority for designing the tool has focused on a very straight class design. The concept has been designed with an clear structured interface between the core simulation library and the GUI in order to allow the use of the core library only respectively the use of the tool by the GUI without any change of the libray.

JAVA-SIM CONCEPT

The JAVA-Sim concept is a modern simulator concept, which combines builing block structures, very hight flexibility and the ease of use through on a new control concept. There are already some simulation libraries based on JAVA available, such as e.g. SILK, which is a JAVA-based, process oriented simulation system. However these libraries doesn't fulfill the requirements of the project completely. In order to realize an advanced concept without limitations and compromises given by existing software the decision was taken to develop a completely new library. This JAVA-Sim library includes multible inheritance and a very flexilbe approach. Additionally the development of an comfortable GUI has been started. Further commication and interface modules will follow in future.

The JAVA-Sim concept offers different user levels. On the library level JAVA simulation classes and additionally all features of the JAVA language are available designing a model. This level allows a very flexible modeling and the integration of the classes into other applications. However on this level good JAVA programming knowledge is mandatory. The second level focuses on users from the production planning area. On the second level the user shall be able to develop models by use of a comfortable GUI without programming knowledge. An advanced control concept allows to design and to parametrize very flexible controls without programming. The combination of these levels offer to design applied building blocks for special production areas. Further the multible inheritance of object parameters is an very important feature of the JAVA-Sim concept.

The architecture of the simulation library and the GUI is object oriented. This guarantees the user a clear structured hierarchical approach for reusable and hierarchic components. The GUI allows to built new simulation models with an easy and flexible approach by use use of basic simulation classes and highly integrated applied material flow objects. The graphical user interface offers multilateral possibilities to create special simulation components, with all features of inheritance. The user interface transmits the data, through an interface, to the simulation classes and conceives the simulation data, to be handled in the visualization. The animation of the simulation events is represented in the graphical user interface and can be addionally analyzed by use of an integrated trace functionality.

IMPLEMENTATION

For the implementation of the simulation tool JAVA as a modern programming language has been chosen. JAVA forces to implement a consequent object-oriented architecture. The platform independent implementation in JAVA is open for different operating systems and to integrate modern communication modules. The disadvantage of a slightly reduced processing speed compared to other languages will not be a very important factor in future.

COMFORTABLE USER INTERFACE

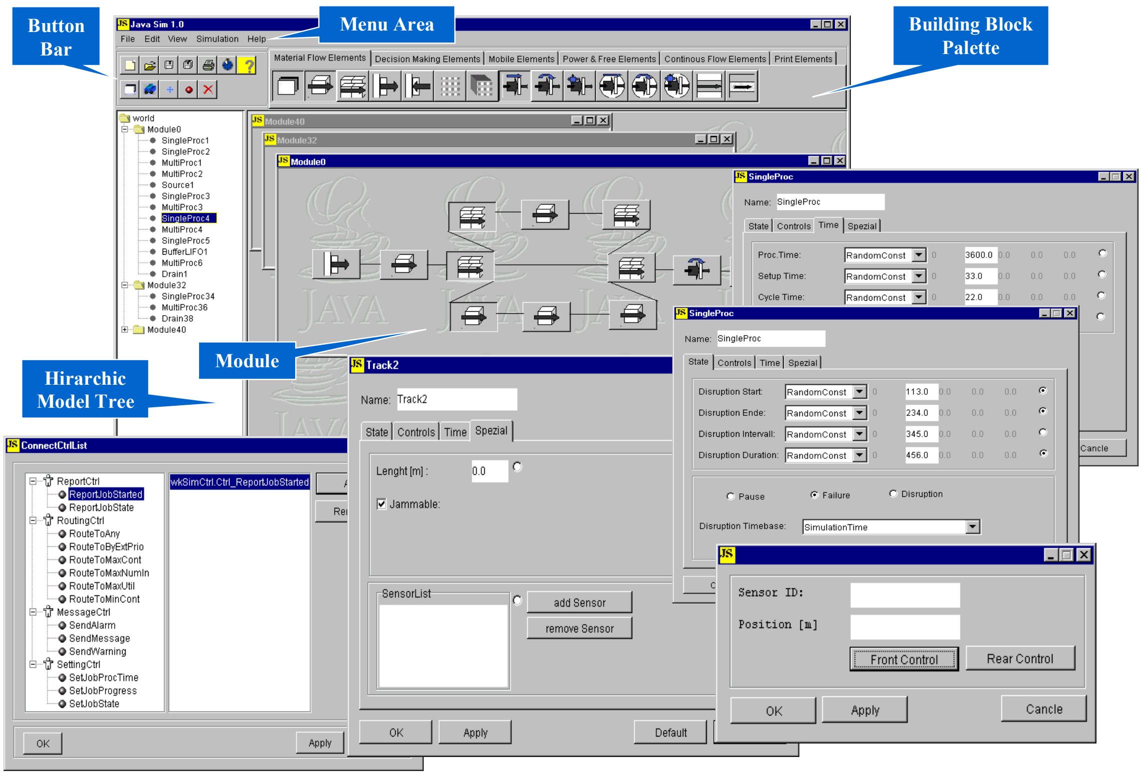
The user interface is structured into the menue bar, the building block library bar, the hierarchical model tree and the desktop area.

Menue bar

The menue bar offers buttons for the most important functions. These are containing self-explaining icons in order to enable the acquaintance with the interface is relativly easy. During use of the program online help is available. While the mouse is resting upon a button a tool tip texts appears and the user gets a short information about the particular button.

Desktop Area

JAVA-Sim provides a distinctive desktop area in which all modules can be shown. JAVA-Sim operates with a single main window and many simultaneously open sub-windows, which can be set within the mainwindow as desired. This technique,



world
- Module0
  - SingleProc1
  - SingleProc2
  - MultiProc1
  - MultiProc2
  - Source1
  - SingleProc3
  - MultiProc3
  - SingleProc4
  - MultiProc4
  - SingleProc5
  - BufferLIFO1
  - MultiProc6
  - Drain1
- Module32
  - SingleProc34
  - MultiProc36
  - Drain38
- Module40

Name: SingleProc

Proc.Time:	RandomConst	3600.0
Setup Time:	RandomConst	33.0
Cycle Time:	RandomConst	22.0

Name: SingleProc

Disruption Start:	RandomConst	113.0
Disruption Ende:	RandomConst	234.0
Disruption Intervall:	RandomConst	345.0
Disruption Duration:	RandomConst	456.0

Pause [✓ Failure] Disruption

Disruption Timebase: SimulationTime

Name: Track2

Lenght [m] : 0.0

[✓] Jammable:

SensorList

Sensor ID:

Position [m]

- ReportCtrl
  - ReportJobStarted
  - ReportJobState
- RoutingCtrl
  - RouteToAny
  - RouteToByExtPrio
  - RouteToMaxCont
  - RouteToMaxNumIn
  - RouteToMaxUtil
  - RouteToMinCont
- MessageCtrl
  - SendAlarm
  - SendMessage
  - SendWarning
- SettingCtrl
  - SetJobProcTime
  - SetJobProgress
  - SetJobState

wkSimCtrl.Ctrl_ReportJobStarted

Figure 1: JAVA-Sim GUI

known under Windows as MDI (Multiple Document Interface) has been chosen to enable a drag & drop functionality for the building blocks. Further this system is relatively easy to administrate.

Building Block Library Bar

The building block library bar is another central element. By use of a wide building block palette with basic building blocks, predefined standard simulation elements and applied building blocks the effort for designing a simulation model can be greatly reduced. Compared to building block concepts of the 70's the flexibility has been increased considerably and there is the powerfull feature available to design on a lower level applied building blocks for a particular application area tailored due to the customers needs.

Hierarchic Model Tree

For a comfortable overview of complete projects, respectively some parts of a project, a confortable model tree is provided, which gives an hierarchic view of the model. The navigation within the model is very easy and a fast interaction with certain areas is possible and the abilities to monitor and understand the model are increased considerably.. Modules are represented as folders as long as these contain other building blocks. A simple double-click opens a module and shows the user its content. A double-click onto the module opens its module frame if it is not open already. A double-click on any another simulation object within the hierarchic view opens the corresponding object modelling window, which allows the user to parametize the object to the particular needs.

Object Modelling Window

Simulation objects in JAVA-Sim are predefined, highly developed and specific building blocks. In these building blocks the basic behaviour and available functions are represented. Theses can be inherited, duplicated and combined as often as desired. Through a special object modelling window each building block can be individually parametrized in order to establish the requested behaviour, which might differ from the basic behaviour. This allows to create realistic models of real system as required. The parameter setting can be modified during the simulation modelling by the user. A double click onto a building block within a module, or it's equivalent in the hierarchic model tree opens the simulation object input window with the actual parameter data view.

The object modelling window is generally structured into the register cards *State*, *Control* and *Time*. These represent the standard possibilities of modifications of the material-flow objects. If certain simulation objects are not able to support some of these data, those are are locked and shown under laid with grey. For special data each simulation object has an additional register card labelled *Special* in which the building block specific settings can be defined.

JAVA-SIM SIMULATION LIBRARY CONCEPT

The JAVA-Sim Library is a simulation library, developed for the simulation of discrete event processes. For the implementation of the library JAVA has been chosen. JAVA is a modern programming language and can be used to implement a consequent object-oriented architecture. An important precondition is that the programming environment does not use platform specific features. For the library a consequent structure is required. Therefore the priority has been focused on a straight class design and not on powerful animation features in the first state.

The JAVA-Sim Library is not limited on a certain simulation level like many available simulation tools. The architecture of the library uses an object oriented approach and is very open and modular. The library shall offer features similar to the flexibility of a simulation language, through the level of general building blocks, such as workstations and assembly stations, up to the efficiency of simulators with special building blocks, such as e.g. power and free system elements.

The JAVA-Sim library follows an object oriented design. It contains public simulation classes, made available for the user of the simulation library, and internal simulation classes and utility classes for internal use in the library only. The library offers general simulation objects, material flow objects, information flow objects, decision making objects and applied material flow objects. The internal class InfFlowObject is used to implement information flow. Special information flow elements may be derived from this class. The class SimEventObject is based on the class GeneralObject and contains the additional functionality for creating simulation events. From this class the general classes EventGenerator and MathFlowObject and MathFlowObjecExt are derived. Again from these the detailed material flow classes such as Buffer, BufferFifo, BufferLifo etc. are derived. For all simulation classes there are some general class features available, e.g. such as create object, delete object, constructor control and destructor control.

GENERAL SIMULATION OBJECTS

The general simulation classes offer common features required for a simulation run. General simulation classes are the SimEnvironment, and the GeneralSimModel with the SimController.

The SimController is the simulation engine of the discrete event simulation. It includes the function of event tracing and event debugging. The SimController has to handle all events taking place during a simulation run. Therefore the SimController has an EventList, which contains all actual simulation events. The methods of the SimController have to deal with incoming events, with the sorting of events and to calculate the event statistics. The SimController offers methods for parameter setting, methods for running the simulation and methods for getting information from the SimController.

The EventTracer is part of the SimController. If the EventTracer is activated, it writes a trace of all or selected events during the simulation. This trace can be saved in a file and may be used to analyze the simulation run later in detail. After validation of a model the EventTracer may be deactivated in order to speed up the simulation run.

The EventDebugger is also an optional part of the SimController and provides an advanced control of the execution of simulation events. A breakpoint flag can be set on certain events or event types. The EventDebugger may be used to analyze a simulation model step by step during the modeling and validation phase.

The GeneralSimModel is a basic object for building simulation models. The GeneralSimModel contains always one SimController, which handles the events occurring during a simulation run, and one ModelStructure, which contains all information about the actual model. For the ease of use special simulation model classes with additional features may be derived from the GeneralSimModel class. The ModelStructure contains the information of the model hierarchy and the model elements. All elements of a particular model are listed in the structure list.

The Module is a general object which serves for grouping objects and building hierarchical structured models. The Module does not have specific basic properties.

SIMULATION EVENT OBJECTS

Simulation event objects are non material flow objects creating active events containing no material flow. Examples for non material flow simulation event objects are e.g. the EventGenerator the Shift or the Scheduler.

The EventGenerator produces events randomly or at fixed intervals. Therefore the start of event generation, the interval between events, the duration of events and the end of event generation can be defined using random distributions or constant values. The EventGenerator can be used to trigger actions during a simulation run.

The object ShiftCalendar is used to define different shifts for the model. The ShiftCalendar contains a Shift object for each particular shift. In the ShiftCalendar additional working days and non working days with a change of shift can be defined. The Shift object keeps the information about a particular shift, such as the start and end of shift and the times for breaks. All resources using this shift are listed in a resource list.

The Scheduler is an active event object containing a list of future events sorted in a defined order. The Scheduler offers to control and administer planned events. If a new event is added to the scheduler the scheduler will sort the list automatically.

MATERIAL FLOW OBJECTS

Material flow elements can be classified into stationary and mobile material flow objects. Stationary objects represent mainly the available resources in a factory, mobile objects the material flow through the factory. Further material flow elements are classified into active and passive material flow elements. Active material flow elements are able to pass the ME’s actively to a next element, on the passive material flow elements the ME’s have to be active elements or have to be moved by an external control. Active material flow objects are the Source and the Drain, the SingleProc and the MultiProc, the Conveyor, the Sorter, and the Buffer. Passive material flow elements are e.g. the Warehouse and the Track which do not automatically pass ME's to a next location.

Mobil material flow elements, such as the Transporter, the Container or the Part represent the physical or logical elements moving through a model. Mobil elements require stationary material flow objects, such as a SingleProc, a MultiProc, a Buffer or a Tracks to be moved on.

Stationary material flow objects are further classified in place oriented and length oriented elements. Capacity oriented material flow elements, such as the SingleProc, the MultiProc and the Buffer, may contain a certain number of elements independent from the size of each element. Length oriented elements, such as the Conveyor and the Track, take the length of the stationary object and the mobile element into account.

In general all material flow objects have some common properties, such as methods for the state, and additional some object specific properties described in the following.

PLACE ORIENTED ACTIVE MATERIAL FLOW ELEMENTS

Place oriented active material flow elements are the SingleProc, the Source, the Drain, the MultiProc, the Buffer and the BufferZeroTime and the Sorter.

The SingleProc is an active material flow object. It has a single station for processing one mobile material flow objects (ME) at a time. The SingleProc sends the ME after passing the set-up and processing times towards the next material flow object. While a ME is located on the SingleProc, the SingleProc can not receive any additional ME. The SingleProc is a place oriented material flow object, therefore the ME is passed always completely onto respectively out of the SingleProc.

The Source is a single place station with a capacity of one and no processing time. The task of the Source is to create mobile elements (ME's). The Source offers methods to define which mobile elements (ME's) and how often these shall be generated. The source is an active material flow object and tries to move the generated ME to the next connected material flow object.

The Drain is a single place station with a capacity of one and no processing time. The task of the Drain is to delete mobile elements (ME's). The parts deleted can be noted in a protocol list.

The MultiProc has several stations for processing mobile elements (ME's). The basic properties of the MultiProc are the same as those of a SingleProc, however with multible processing stations parallel or in line. A set-up time is always applied when an ME has a different setup-type than its predecessor. Due to the place oriented characteristic of the MultiProc material flow element are always passed completely.

The Buffer has a definable queue length for mobile elements (ME's). The basic properties of the Buffer are the same as those of a SingleProc with several processing stations in line. The Buffer is a place oriented material flow object and has several processing stations in a queue. ME's are processed from one station to the next. Further Special buffers are the BufferZeroTime and the Sorter

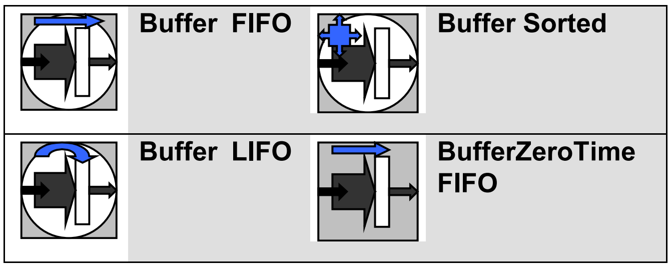


Figure 2:   Some Buffer Types

LENGTH ORIENTED MATERIAL FLOW OBJECTS

In length oriented material flow objects the length of the stationary material flow object and the length of the mobile element are considered for the simulation calculation. The Conveyor and the Track, are basic length oriented material flow objects, the PowerAndFreeTrack is an example for an applied length oriented material flow element.

The Conveyor is a length oriented material flow object. The Conveyor can be used to model length oriented transport systems. The Conveyor transports ME's along its entire length with a constant speed. On a conveyor ME's cannot pass each other. If a ME is located on more than one Conveyor for the calculation the speed of the Conveyor is used where the booking point of the ME is located on.

The Track is a length oriented passive material flow element. The Track is used for modeling transport lines. The Track requires active mobile elements, such as the Transporter or the AGV (automatically guided vehicle). The required time of a Transporter on the Track is calculated by the distance it has to travel and its speed. ME's may not pass each other on the Track. The capacity of the Track may limit the number of ME's staying on the track at the same time.

MOBIL MATERIAL FLOW ELEMENTS

The mobile material flow objects are not fixed at a certain location. Mobile elements are passed along the material flow from object to object. During a simulation run these elements move through the model and represent the discrete material flow. Mobile material flow objects offered by the JAVA-Sim Library are e.g. the Part, the Container, the Transporter the AGV and the PowerAndFreeHanger.

The Part is a mobile material flow object without any loading capacity. The Part represents parts being produced and transported. Due to passive element characteristics of the Part it does not have active properties. Therefore it has to be moved by an active material flow element or by an control.

The Container is a mobile and passive material flow object with definable loading capacity for the transport of other ME's. The capacity is place oriented, referred to a defined number of ME's only. Note that the Container does not take the physical size of the loaded ME's into consideration. The Container can be used to represent palettes or boxes.

The Transporter is an active mobile material flow object. It has a loading capacity for transporting mobile elements, such as Entities, Containers or other Transporters.

INFORMATION FLOW OBJECTS

The JAVA-Sim Library offers basic information flow objects described below. These can be classified into passive objects for keeping data, such as the global DataStore and one and two dimensional lists, such as the ListX, the ListLIFO, the ListFIFO, the ListXY. Another classification are active control objects, such as the ControlMethod. The class random distributions offers the required random values for various parameter settings.

JAVA-Sim Library offers the objects ListX, ListFIFO and ListLIFO as lists with one column for storing data. The objects ListFIFO is a list sorted by FIFO (First In First Out) and ListLIFO is sorted by LIFO (Last In First Out). The ListX is a one-dimensional list with the possibility to access to each individual entry by addressing the position. The ListXY is a list with several columns. It is possible to access the individual entries by addressing its index.

Standard controls are integrated in many material flow objects. The JAVA-Sim Library offers the ControlMethod object for applying special controls. This user definable controls can be inserted into various objects in order to control the material and information flow in the required manner.

The JAVA-SIM Library supplies a number of statistical distributions, such as the uniform distribution, the triangular distribution, the normal distribution, the lognormal distribution or empirical distributions. Various object specific parameters, such as ProcTime, SetUpTime or in the Disruption object the FailureDuration or FailureIntervall may use values according to these distributions.

The JAVA-Sim Library offers several decision-making objects. These decision making objects can be classified into objects directly connecting material flow objects, such as the Connector and the ConnectorN2M, and objects which are not integrated directly into the material flow, such as the ServicePortal, and objects to be embedded into material flow objects such as the ServiceProvider and the ServiceRequester.

For connecting material flow objects and for the routing control between material flow objects the JAVA-SIM Library offers the Connector, the ConnectorN2M and the RoutingCtrl.

The Connector and the ConnectorN2M are object for connecting material flow objects. The ConnectoN2M provides features to decide how to route the mobile elements through the system.

The RoutingCtrl is a control object applying strategies for converging and diverging material flow. The RoutingCtrl offers some standard controls for material flow routing as well as the possibility to implement user-defined strategies.

APPLIED MATERIAL FLOW OBJECTS

The Applied Material Flow Objects are material flow objects for specific applications. These objects are based on the basic material flow objects defined above. Some of these applied material flow objects are kept very general, such as e.g. the WorkStation, other applied material flow objects are very specific for certain applications, such as e.g. the objects for modeling Power & Free systems.

SUMMARY

The JAVA-Sim concept allows to run simulation models on various hardware platforms and operation systems. The platform independent implementation in JAVA allows a use of the library on different operating systems and to integrate modern communication modules. Disadvantage of using JAVA is a slightly reduced processing speed compared e.g. with C++. However with the performance increase of the hardware this will be not a very important factor in future. With help of the developed GUI the user shall be able to design models without knowledge of the JAVA programming language, even if complex control structures are required.

REFERENCES

Burke, E. and R. Kilgore. 2000. Silk®, JAVA AND OBJECT-ORIENTED simulation. proceedings of the 2000 Winter Computer Simulation Conference. SCS International, Ghent, Belgium.

Kühn, W., JAVA-Sim - An advanced Discrete Event Simulation Library, SCSC 2002, 2002 Summer Computer Simulation Conference, San Diego

BIOGRAPHY

SUSANNE OSE studied print and media technologies with a focus on production planning and control at the University of Wuppertal, Germany. In her master thesis she developed a GUI for the JAVA-SIM simulation library.

WOLFGANG KUEHN studied mechanical engineering at the University of Brunswik, Germany. Afterwards he worked two years with Blaupunkt. At the University of Bremen he got 1991 his PHD in production engineering and 1997 his habilitation in the area of simulation of production systems. From 1993 to 1995 he joined the Asian Institute of Technology in Bangkok as an Associated Professor. 1996 he founded the SIPOC Simulation based Planning, Optimization and Control GmbH in Bremen and since 1997 he has a professorship for production planning and control at the University of Wuppertal.