OSD CASE-tool for Distributed Real-time Systems

BOGOMOLOV S., BERDNIKOV A., FYODOROV A., KUDRYAVTSEV D.
LG Technology Center of Moscow
St.-Petersburg’s branch, Soft Lab
Shpalernaya str. 36, 191123 St.-Petersburg
RUSSIA

Abstract: The description of the first version of CASE-tool designed in LGTCM Soft Lab and named OSD is presented. It intends for development of real-time embedded and telecommunication systems, and other systems that are reactive, distributed and real-time. Software developer can specify a visual model of future system using graphical notation based on UML, SDL and MSC International Standards supporting analysis and design stages. The simulation is used for preliminary verification of the specified system. To simplify debugging during simulation mode OSD-tool uses a visualizer which provides quick access to variables for the given context and shows the appearance of events on time scale during communication between processes, the current status of processes, etc. After debugging the developer can be able to generate C++ code to insert into target platforms. To provide compatibility with other CASE-tools there is an OSD-converter. Development platform works under Windows 98, NT. Target platforms are VxWorks, Windows NT.

Key-Words: CASE-tool, real time systems, simulation, code generation, UML, SDL, MSC standards

1 Introduction
Real-time systems are used in a wide area of industries such as telecommunication, aviation, military control systems, atomic power stations, automotive, etc. The development and maintenance of software for large-scale distributed real-time systems is highly complex. It demands huge efforts from developers to provide high reliability and sound communication of distributed components located on heterogeneous platforms, satisfy critical time constraints. In one’s turn the market demands to reduce time-to-market and provide constant functionality upgrading.

CASE-technology (computer aided software engineering) is one of key ways to overcome such kind of difficulties. It is based on:

- support of the entire development lifecycle including analysis, design, implementation and testing through visual modeling, simulation and code generation,
- using international standards for graphical specification of real-time systems (SDL, MSC from ITU-T and UML from OMG),
- debugging on the earliest development stages using simulation and visualization of graphical specification behavior on host and target platforms.

Owner of CASE-tool gets obvious advantages in the market at the expense of:

- reducing of software development time,
- less costly development,
- easier updating,
- better cross platform support,
- increasing of designed system quality.

The most known commercial CASE-tools for development of real-time system software are ObjectGeode (Verilog, France), SDT (Telelogic, Sweden). They are very expensive and insufficient to customize needs of developers. There is an insufficient integration of UML analysis and SDL/MSC design stages. Known Rational Rose package (Rational, USA) supports analysis stage but is not connected with SDL/MSC design stage.

In this paper a brief description of the first version of OSD CASE-tool, its functionality and perspectives are presented.

2 OSD-tool editors
OSD-tool intends for development of real-time embedded and telecommunication systems, and other systems that are reactive, concurrent, distributed, real-time, heterogeneous and complex.

OSD-tool includes several editors supporting UML, SDL and MSC-modeling. Corresponding editors are tied together, so whenever you make changes in one they are immediately shown in the others. User can invoke editor for a component from the Workspace or any other editor window.
containing the component. Every component drawn with any of SDL or UML graphical editor is shape. Last created component shape lays over all another component shapes. Sometimes it is convenient to change shapes order. OSD allow user to move component shape back and forward. It gives ability to create any shapes order.

The common view of OSD–tool is presented in Fig. 1. On the left side there is Workspace Window which has several tabs for user’s convenience:
- **SDL Specification**,
- **TestCase and MSC generation**,
- **UML Structure**,
- **InfoView**.

InfoView tab provides easy access to OSD online documentation and help. **SDL Specification** serves for SDL - editing. **TestCase** is used to select a part of SDL-model for following simulation and MSC diagram generation. **UML Structure** intends for UML - specification.

**SDL-s**pecification is used both of graphical and textual representation and allowing to specify by graphical way the following elements: system, blocks, channels between blocks, communication between components using signals carried by channels, gates and routes, behavior including processes as extended communicating state machines, procedures and services, data including predefined data types and ASN sequence Of construct as substitution of SDL predefined Arrays and Strings. Additionally to standard SDL specification, the user is able to use built-in probability distributions to specify probability of spontaneous transitions and channel delays.

**Specification** tab lets the user access a project (system being under design) structure. Every item in the object tree has a context menu associated with it.

To bring up context menu for a given item the user can right-click the item in the workspace tree. Popup menu allows to:
- add selected module to **TestCases Window** (see below) for simulation,
- create the type by an instance,
- create the instance with the selected type,
- add corresponding component to selected module,
- open graphical SDL-editor for the selected component,
- remove the module and all its components,
- change the specific properties of the selected object which can be accessed via Property Pages.

**TestCases** Tab of the Workspace Window serves for the following purposes:
- specifying and managing test cases during system development,
- managing processes during simulation,
- MSC diagram generation after simulation.

One of the test cases can be selected and after be interpreted when simulation starts.

**UML Structure** tab in Workspace window lets a user access UML-structure and UML-editors. The following UML diagrams editors are supported in OSD-tool: Use Case Diagram, State Chart Diagram, Class Diagram, Component Diagram, Deployment Diagram. The diagrams can be used for analysis stage of software development cycle and conform to International Standard UML 1.1.

Message sequence diagram (MSC) represents an interaction, which is the set of messages exchanged among objects within a collaboration to effect a desired operation or result. A message sequence diagram has two dimensions: the vertical dimension represents time, the horizontal dimension represents different objects. Normally time proceeds down the page. Usually only one time sequences are important but in real-time applications the time axis could be an actual metric. MSC diagrams in OSD-tool are used for editing, Test cases generation, Execution logging.

## 3 Simulation in OSD-tool

After the system specification is finished and its consistency is checked the user is able to verify and debug the system in simulation mode. Simulation is used for preliminary analysis of the application correctness and estimation of its performance. In the simulation mode the most detailed information on all objects, constituting the system, is available OSD various windows.

Using automation the developer can program OSD to run multiple simulation sessions to find optimal parameter values or to test the application scalability. And all that is performed on a single developer's machine.

Except continuous simulation mode called **Run**, user can simulate the system in step-by-step mode called **Step**. In this mode step means the one action (task, input, output, etc.) performed by a process. In this mode the following steps can be done:
1. Step - one step of one of processes is executed
2. State - selected (by the simulation engine) process is executed until it enters a state
3. Process Step/State - same as above, but for user-selected process only (step for other processes are executed without interruption)
User is able to choose between simulation with or without animation, run the system to time/step, exist also a lot of another possibilities.

OSD simulates the diagrams not in real, but in virtual time, thus making arbitrary complex experiments possible on a single workstation. When you start the model executable, the built-in simulation engine creates and initializes the model, the event queue, the model clock and the active object tree. Depending on the presence of events in the event queue and the error status, the model comes to either Step by Step or Finished state and the control is passed to the user.

4 Visualization in OSD-tool

Visualization is achieved by use of the project database and various views (e.g. graphical editors in read-only mode). Fig.1 shows an example of visualization view of system during simulation.

The aim of visualization is to show dynamic situation on Developer's platform (simulation) and on Target platform (target execution). However the control over the execution and the amount of information available to the user in these modes will be different. For example, in the simulation mode the user can watch the current states of the processes’ objects, whereas in the real mode these details are not available.

Visualization is aimed to give user information about:
- current statuses of the processes,
- appearance of events for given process,
- changes of a variable's value from given variable list.

5 Code generation in OSD-tool

The code generation includes two parts. The first one is a platform-independent part which is a text of program on C++ programming language which is generated from visual SDL specifications debugged on the simulation step. It is sited in the folders including make-file, a set of header-files and a set of implementation files.

The second part is a platform-dependent part. It includes the run-time libraries for the given platforms: VxWorks or Windows NT. The library is a set of functions which provides embedding the generated code into the given platform. The initial text of the libraries will be given. On the base of these libraries the developer-user can develop own library for other platforms. The generated code is compiled and linked with the run-time library to have executable code. The text of C++ generated code is appeared in the log called Generation. To generate C++text it is used 'C++'button in toolbar.

6 Customization and compatibility with other CASE-tools

User freely can customize application interface. All the customization goes through customize dialog. With this dialog user can create, reset, rename, hide and delete toolbars, add or remove buttons to toolbars, define accelerators, add macros, etc.

To provide compatibility OSD with other commercial CASE-tools there is necessary to support a conversion from graphical description into standard text description called SDL/PR and vice versa into SDL/GR. SDL/GR => SDL/PR conversion is supported in the current version of OSD-tool. This converter is implemented as macros, so it is convenient to add button on toolbar, which will be convert currently open SDL graphic representation project to SDL textual representation. SDL/PR =>SDL/GR conversion is under design.

7 Conclusion

The main features of OSD-tool as CASE-tool are the following:
- convenient means of customization,
- convenient visualization during simulation supporting various views of behavior and interactions observation especially event diagram window, process and variable windows,
- opportunity to create, add specific macros using Visual Basic Script, provide their access to all project data and run them from OSD development environment together with SDL-model,
- a support of UML, SDL, MSC in one integrated development environment.

The development platform used for design, simulation and visualization can run under Windows 98, Windows NT 4.0. The target platform used for goal program execution will be able to work under VxWorks, Windows NT 4.0.
Fig. 1. General view of OSD-tool.