OVOPS - an Object Oriented Implementation Framework for Protocol Engineering

Jarmo Harju\textsuperscript{a),} Bilhanan Silverajan\textsuperscript{b)}

harju@cs.tut.fi, bilhanan@lut.fi

\textsuperscript{a)}Tampere University of Technology, Hermiankatu 12 C, FIN-33720 Tampere, Finland
\textsuperscript{b)}Lappeenranta University of Technology, Skinnarilankatu, FIN-53850 Lappeenranta, Finland

1. OVOPS in a Nutshell

OVOPS supports the design, implementation and prototyping of protocols and distributed applications by providing an object oriented framework with class libraries and tools that are often needed in the development of communications software. The basic structure of OVOPS is described in Fig. 1. With OVOPS, implementations can be made largely independent of the operating system.

In short, the basic services and tools provided by OVOPS libraries include:

- scheduling of tasks,
- message passing between tasks,
- interfaces to the operating system services,
- graphical and textual protocol tracers for debugging,
- hierarchical symbol interface to support user interactions in symbolic form,
- efficient and controllable memory management,
- multi-threaded support,
- timers, frames and other useful classes.

\textbf{Fig. 1. The general OVOPS model}

2. Protocol support library

The OVOPS Protocol Toolbox (PTB) supplies code generators to produce OVOPS specific C++ code from a set of specifications, which are the following:

- \textit{SAP definition}. The service access points are described and compiled into class libraries for the utilisation of protocol layers.
- \textit{PDU definition}. The protocol data units of the layer are described and declarations of the PDU classes with necessary data structures are generated from these definitions.
- \textit{State machine definition}. The structure of the state machine of a protocol is described and the state machine class for executing the corresponding handling functions is generated from these specifications. Description at this level is rather simple, leaving the details of the actions to be carried out by the handling functions, which are manually coded in C++.

3. Experiences and availability

One area in which OVOPS has been used is the implementation of the core elements of a service control point (SCP) of an intelligent network system. More details are described in Fig. 2. The SSP simulator and the SLPs were provided as external elements. Communication between SCP and SSP uses TCP/IP and the BSD socket interface.

\textbf{Fig. 2. Test implementation environment}

OVOPS has been ported to Solaris, HP-UX and Linux operating systems with GNU C++ 2.7.2 and the complete source code and documentation is freely available at

http://www.lut.fi/tikk/tite/labs/dc/ovops/