Web-based system for learning of communication protocols

Dan Komosny

Brno University of Technology, Czech Republic

Summary
The paper introduces a new web-based system that provides on-line access to software source codes developed using the Specification and Description Language (SDL). The purpose of this web system is to support the teaching of protocol techniques by offering practical examples of protocol implementation. However, examples developed in low-level languages such as C++ do not clearly describe the communication process as is desired. Therefore, a formal approach is often used for teaching purposes. SDL is one of the well-known formal languages used. SDL is an object-oriented programming language that uses graphical expressions for the source code. Furthermore, a simulation and validation of the SDL software can be performed without real implementation. The simulation results are again presented graphically using the Message Sequence Charts (MSC) diagrams. The above web system offers users the possibility to publish and search through SDL source codes with accompanying MSC diagrams.

Key-words:
protocol, e-learning, SDL, MSC

1 Introduction

The teaching of communication protocols has usually two forms - theoretical and practical. The practical form usually involves developing a selected protocol part using a common programming language. However, the commonly used low-level languages such as C++, Java or C# do not allow students to easily understand the protocol structure and its behavior. The problem is that with these languages, the specification and implementation are quite different. Moreover, the communication process is usually hidden in a huge amount of program code, including parts for encapsulating interfaces and operating system specific functions. This makes the final program code difficult to understand and, consequently, it is not very suitable for teaching the networking techniques. On the other hand, teachers often use manually drawn flow-state diagrams in order to show the basic behavior of a communication protocol and its internal messaging. In some cases, this does not give the intended information as desired and, furthermore, such diagrams quite often do not relate to a real system implementation. Therefore, these teaching techniques do not give a complex overview of the protocol presented - they are either too general or too concrete.

A solution to this problem can be found in using a formal specification language [1]. A formal approach can be useful in a number of teaching ways. Formal description is also of central importance in standardization. Networking protocols are often specified in this language. Generally, formal languages are used because they make the developers think in detail about the system at an early stage of the development. This helps to create an error-free system on the first go. The higher-layer abstract notations included in the formal approach allow designing in preliminary stages of system development. All this results in abstract design instead of the coding from the early beginning. The abstract approach is mainly significant in complex systems. Developers or customers can have an overall overview of the system and they can go into detail in partial parts. Several techniques for formal specification have been standardized, such as LOTOS (uses a process algebra), Estelle (uses finite state machines) and SDL (uses finite state machines), see [2] [3]. Fig. 1 shows an overview of these formal languages. More specification oriented approaches are thought to be more abstract without definition of specific behavior (i.e. limited timing models, verifiable, incomplete language with limited data parts, declarative language styles and static process structure). On the other hand, more implementation-oriented approaches are thought to describe specific behavior (rich model of time, validation through simulation, complete language with usable data parts, imperative language styles, dynamic reconfigurability).

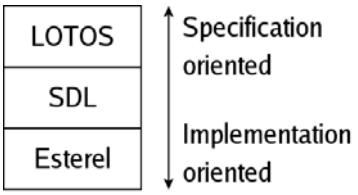

Fig. 1 Formal specification techniques

2 Specification and Description Language

From the above languages, SDL is thought to have features which comply with the teaching needs. SDL is an object-oriented programming language that uses graphical expressions for the source code. It is a high-level language

(in contrast to the low-level C++, Java ...). SDL is intended to be used for the description of real-time communication systems. It allows system specification on various levels of abstraction, starting from the general view of the whole system and finishing with details, e.g. frame structures. Together with SDL the MSC language is often used. MSC is used for the description of real-time communication behavior. It uses a graphical interface with a textual description of the transmitted signals. MSC is often used as a simulation outcome of an SDL system. Therefore, using MSC diagrams, the signal transmission can be followed and the system behavior checked easily. SDL together with MSC can be seen in many communication fields such as cellular phones, Bluetooth devices, GPRS systems, DECT phones, radio systems, network services systems, and so on.

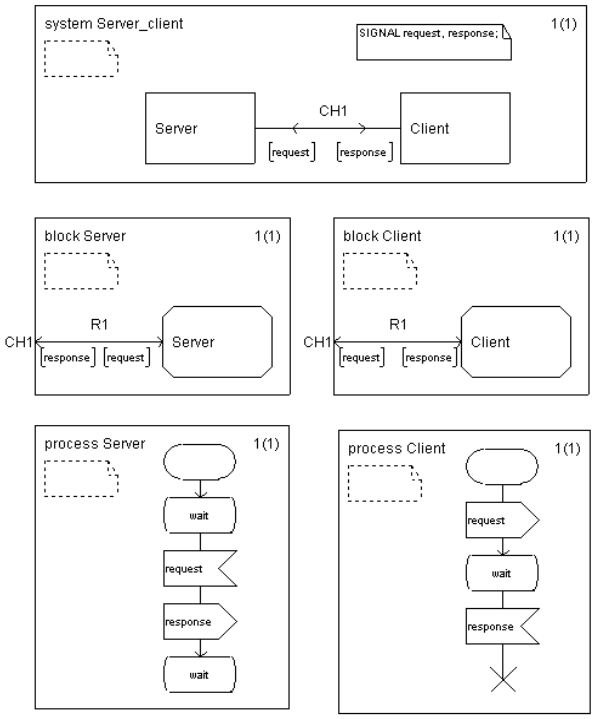

Fig. 2 Server-client system example

A simple overview of SDL is depicted in Fig. 2. The client-server system shown allows the transmission of request and response signals (packets). The system consists of various levels of abstraction. The high level represents the general communication, i.e. the server-client channel; the lowest level represents more detailed structures, for example a state machine. Each process in the SDL is represented by a state-extended finite machine, which describes its dynamic behavior. These finite state machines can have timers and variables as in classic languages. Indeed, an SDL model involves lots of state machines. Fig. 3 shows a part of the communication using MSC. It can be seen that both SDL and MSC are closely related to each other. For further information about software development in SDL see [2].

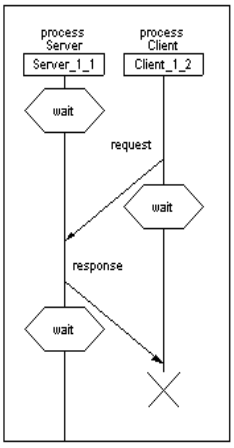

Fig. 3 Server-client example

Development environments for formal languages usually also include some proof and validation techniques. The well-known is the Telelogic TAU SDL Suite [4]. It has several handy features such as the simulation and validation of formal characteristics, code compilation into the C++ low-level language (i.e. a formal description can be automatically changed into an executable application), and simple integration with MSC. A project developed using this suite can be performed without an implementation in the target system. This allows building hardware-independent applications.

Fig. 4 shows the development process from the initial idea to the final application. Steps 1-3 covering the idea notation, design proposal and final software implementation are all made using the SDL. Then, a transformation into a common low-level language is processed. This allows running the final application on devices where the C++ language is supported. The final step, displayed using MSC, is the simulation and validation. Usually, there procedures are specific to the developing environment used.

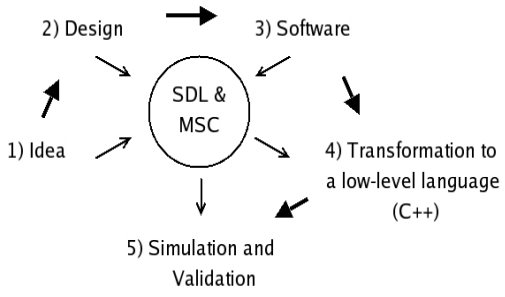

Fig. 4 Use of SDL and MSC

3 Web system for SDL

The web system developed is intended to provide free Internet access to sample program codes created using SDL. The system allows SDL from Telelogic TAU SDL Suite to be easily uploaded to the web server. Then, project viewers can see the graphical structure of a communication protocol displayed using SDL and the system dynamic behavior shown using the accompanying MSC diagrams. An SDL project is divided into elementary parts (blocks, processes, charts). These parts are linked together using web links. Therefore, the user can easily go through the software from the uppermost layers to the lower ones. This feature is helpful if someone is only interested either in a general overview of the software or in a particular part. As the web system is freely accessible, any SDL developer can upload their projects and finally make a text description of the project to help understanding the protocol. Also, projects can be downloaded and used off-line. The web server is available at the address http://sdl.utko.feec.vutbr.cz/project_list.php. The main window of the web system is shown in Fig. 5.

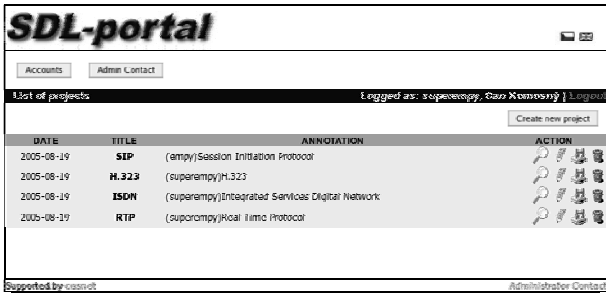

Fig. 5 Main window of web system

When a developer would like an SDL project upload to the server, the following steps are processed. After the development of a system using the Telelogic Tau development environment, the SDL source code is checked for errors (analyse) and then an .exe file is made (make). Then, the developed system can be simulated in order to create MSC diagrams describing the dynamic behavior. It would be worth creating partial MSC diagrams for error-free behavior, e.g. connection establishment, data transfer and disengagement including various possible scenarios. Furthermore, partial diagrams could be made for error states such as packet lost, packet time-out, or receiver/server malfunction. Each of the MSC diagrams created should be added to the project file list for further processing. Eventually, all project files are printed using an integrated print tool in the Telelogic Tau environment to create .png files encapsulated into web pages. That is the final outcome from the development environment.

The next step is to compress the outcome files into one .zip file. Then, using a web uploading tool (see Fig. 6), the .zip file is uploaded to the web server. The system automatically decompresses the .zip file into the structure for web publishing. After this step, the owner of the project can add the formatted text description (see Fig. 7) to each part of the developed system. The purpose of text notes is to give a description of software parts to support easy understanding of the protocol structure and its behavior. When the system description is finished, the system is ready to be published. Finally, any user visiting the web pages can see the system structure presented as SDL diagrams and also various system behavior scenarios

Fig. 6 New project upload

presented as MSC diagrams.

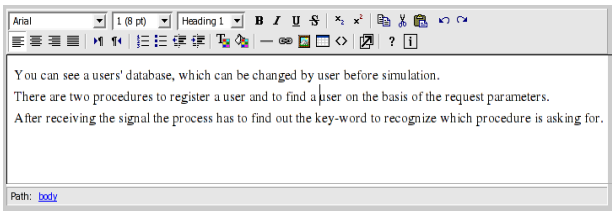

Fig. 7 Creating a notice in text editor

4 Posted software

To initially support the web system, three example communication protocols were developed. These protocols represent basic techniques for multimedia communication in the Internet. Two of them, H.323 [5] and SIP (Session Initiation Protocol) [6], cover signalization for connection management, security issues, and bandwidth control in IP-based multimedia transmissions. The third protocol is RTP/RTCP (Real Time Protocol/ Real Time Control Protocol) [7]. You can seen an overview of these protocols in Fig. 8.

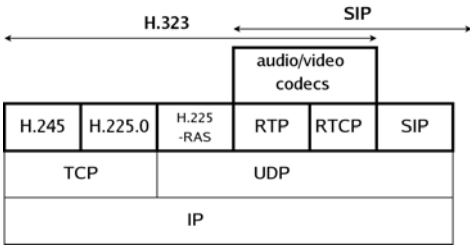

Fig. 8 Overview of protocols

The H.323 Recommendation covers several partial protocols, such as H.225 and H.245. It is a part of the H.32x protocol family containing, among other things, ISDN. H.323 is a complex protocol and tries to standardize everything in its scope. The SIP protocol is more recent than H.323. The SIP messages are formatted as text (H.323 uses binary encoding). The difference is that SIP only standardizes new protocols and not services as H.323 does. Both H.323 and SIP rely on RTP/RTCP, which consists of two protocols, RTP and RTCP. RTP carries multimedia data whereas RTCP carries signalization and synchronization. RTP is a simple protocol covering the multimedia transmitted. RTCP, which is more sophisticated, provides a set of messages exchanged among session users. These messages are used as a feedback for monitoring the session behavior. The feedback flow involves information such as a summary of the data sent, synchronization of different media (audio, video), packet lost, packet delays, and interarrival jitter.

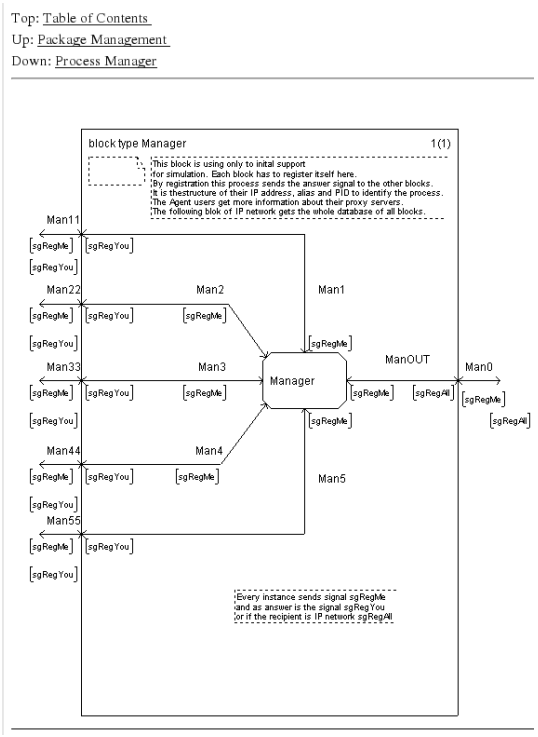

Fig. 9 Block Manager - SIP protocol

For all three protocols, these scenarios were developed for the communication functions: access, transfer and disengagement. Furthermore, in order to show the protocol behavior in certain cases, the packet loss event was implemented. The resulting scenarios show the protocol time-outs with the recovery of the lost packet. An example of the SIP protocol implementation is depicted in Fig. 9 and Fig. 10. The first figure shows a overview of the block Manager. You can see how the block is connected to the environment. An SDL structure of the location process is shown in the second figure. It is a part of the finite state machine including the definition of variables. The process behavior can be intuitively seen through SDL icons and a text descriptions.

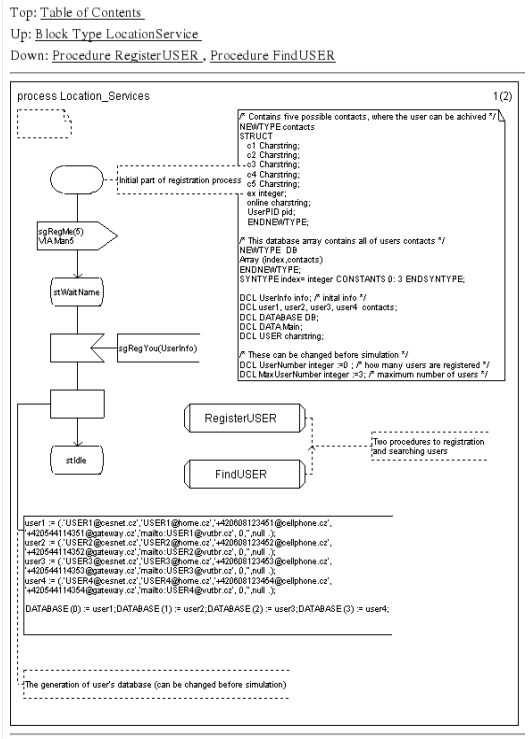

Fig. 10 Process example - SIP protocol

5 Conclusion

In this paper, a web system built for supporting the teaching of communication protocols was introduced. The aim of the system is to offer a graphical interface for on-line access to protocol code examples with accompanying formatted text description. The web system works with SDL formal language source codes. A formal approach to protocol implementation is useful in a number of ways. Instead of using low-level languages such as C++ or Java, a formal language can show much more about the

principle of the communication when a particular protocol is used. Initially, the goal of formal specification languages was to provide a tool for easy specification of protocol standards and for functional analysis to find and correct errors prior to real implementation. However, it could be also used for teaching purposes. Formal languages bring both easy-to-read general overviews of the communication systems and detailed packet structures. Another reason for building the web system is that development environments for formal languages are usually very expensive and therefore they are not affordable for every user. Thus, the formal languages are not much known and most people simply do not know about the advantages they offer.

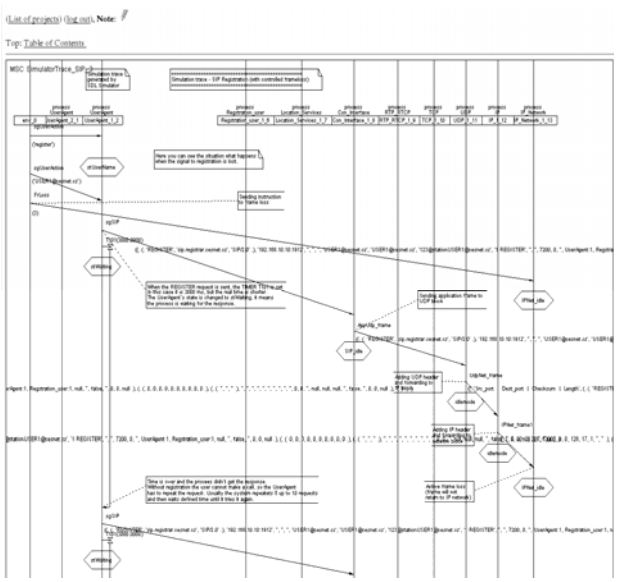

Fig. 11 MSC diagram example - SIP protocol

Acknowledgment

This work was supported by the Academy of Sciences of the Czech Republic (project 1ET301710508) and the CESNET – Czech NREN operator (project 117R1).

References

[1] BLAIR, G., BLAIR, L., BOWMAN, H., CHETWYND A. Formal Specification of Distributed Multimedia Systems. Pretice Hall PTR, 1998.
[2] DOLDI, L. SDL Illustrated. Laurent Doldi, 2001.
[3] SDL Forum Society. What is SDL?. 2003 http://www.sdl-forum.org/SDL/index.htm
[4] Telelogic AB. Telelogic TAU SDL Suite. 2005 http://www.telelogic.com/products/tau/sdl/
[5] KUMAR V., KOPRI M., SENGODAN S. IP telephony with H.323. Wiley Computer Publishing, 2001.
[6] HANDLEY, M., SCHULZRINNE, H., SCHOOLER, E., ROSENBERG, J. SIP: Session Initiation Protocol. Request for Comments 2543, Internet Engineering Task Force, 1999.
[7] SCHULZRINNE, H., CASNER, S., FREDERICK, R., JACOBSON, V. RTP: A Transport Protocol for Real-Time Applications. Request for Comments 3550, Internet Engineering Task Force, 2003.

Dan Komosny graduated from Brno University of Technology, Faculty of Electrical Engineering and Computer Science in the field of Electronics and Communication (2000), Ph.D. (2003). He is engaged in research focused on transmission of voice over IP network (VoIP). He also focuses on development of e-learning tools using formal and visual languages.