The Role of Formal Languages in Engineering Education

Author:

Drago Hercog, University of Ljubljana, Faculty of Electrical Enginering, Tržaška 25, SI-1000 Ljubljana, Slovenia, Drago.Hercog@fe.uni-lj.si

Abstract *— Engineering is not only a mastering of many facts, it is a way of thinking, and engineering education is an education of young people to reason in a specific way. Engineering is an exact science, so that way of thinking must be concise, rigorous and unambiguous. The use of formal languages enh ances rigorous, concise and unambiguous descriptions of engineering concepts and designs, so their use in engineering design, development and education is very important. Moreover, formal languages can help to overcome language barriers by providing for pr ecise and unambiguous descriptions in a well -defined formalism, rather than by complicated intricacies of a natural language syntax and semantics which are not familiar to non -native speakers. This latter fact is especially important in the present time of internationalisation of engineering. Formal specifications of engineering designs are also very important for verification and automatic implementations of designed systems. Since long ago, physicists and engineers have been using mathematical notation to express their reasonings and results. Later on, reasonings and results have been becoming more and more abstract, so the need for more and more abstract formalisms has been arising. Nowadays, a plenty of formal languages exist to describe systems and thei r components at different levels of abstraction and from different viewpoints. The two viewpoints that are most important in engineering design nowadays, are structural and functional viewpoint. A useful formal language must therefore support at least thes e two viewpoints. Teaching a formal language in an engineering curriculum means both teaching useful design and development tools and teaching a way of reasoning. The selection of a formal language to be used in a particular course must therefore be done v ery carefully and with the pedagogic goals of the course in mind. Communication protocols are becoming nowadays more and more important elements of telecommunication theory, practice and education. Their specifications must be concise and unambiguous, so f ormal languages are usually used in specification, design and implementation. In our course on telecommunication protocols, we use the standard language SDL which can support both structural and functional viewpoints at different levels of abstraction. How ever, our pedagogic goals are deeper than only to present an example formalism. We use the notion of an SDL process as a model of a communicating entity. In the frame of this course, each student must specify a simple protocol in SDL and is therefore force d to reason in the way a protocol entity does. In this way, students get acquainted with the operation of systems that function in real time which is one of the most important characteristics of communication systems.*

Index Terms *— Communication protocol, formal language, SDL, way of reasoning.*

INTRODUCTION

The basic role of an engineer is to understand, conceive, design and maintain technical systems and their components. As technical systems differ from other kinds of systems in some of their properties, a future engineer must get used to reason in a technical way, that is in a way similar to the operation of technical systems, with regard to the engineering specialty they are studying.

The most important general properties of technical systems are the following.

- In order to reduce the complexity of the system design and maintenance, systems are designed and built in a modular and structured way, the usual structure being hierarchical.
- Systems are most usually analysed and synthesised from both structural and functional viewpoints.
- Usually, systems operate in real time, which means that their reactions to input stimuli are time-constrained.
- Systems must be functionally correct. Their correctness must be verifiable.
- Systems and their components must be unambiguously specified.

Formal languages are frequently used to specify systems. The following advantages of specification formalisms are usually cited.

- They provide for simple and unambiguous specifications.
- They can be stored, read and interpreted by computers.

- They provide for a basis for computer-aided correctness verification and implementation.
- Being artificial, they are nobody's mother languages, so they are languages of international engineering community.

Actually, formal specifications describe models of analysed/synthesised systems. Such a model must fit the specified system, and the specification language must fit the model. Therefore, different specification languages are used to specify different kinds of systems, and any language chosen must fit the kinds of systems it is intended to specify.

As formal specification languages are regularly used in system analysis and design nowadays, engineering students must of course be acquainted with them during their university studies. However, the most important experience they must gain in this respect is not to know one more language (that most of them will never more use in their practise), but to be forced to think in a structured, formal and unambiguous way which is close to the way that systems they study operate! From this viewpoint, the study of a formal language is a very important component of an engineering curriculum.

In the rest of this paper, the role of the specification language SDL in a communication protocols course will be explained with more details.

COMMUNICATION SYSTEMS

In a communication system, two or more communicating entities communicate among them according to a set of rules called communication protocol. In a complex communication system, communicating entities and communication protocols are organised in a layered structure, usually called protocol stack, where the entities of the same layer virtually communicate among them by exchanging so called protocol messages, while an entity can physically interact with the entities residing directly above and below it by means of messages called primitives. Communication protocols must be unambiguously defined and verifyable for their correctness.

The usual way to specify a communication protocol is to specify the behaviour of a communicating entity and the syntax and semantics of protocol messages to be interchanged between protocol entities. Basically, a protocol entity must be functionally similar to the model shown in Fig. 1 (regardless of the fact that it can be implemented either in hardware or software). Hence, it must possess the functionalities of decision (processor), information storage (memory), and input/output. Furthermore, a protocol entity must be able to measure time (using timers). All inputs and timer expirations must enter the processor through an input queue.

As can be seen from the protocol entity model shown in Fig. 1, the operation of a communication system is time-dependent and hence essentially different from sequential systems, such as classical computer programs whose operation can be shown only as a sequence of basic operations, although many similarities of both kinds of systems also exist. As engineering students usually do have already some basic knowledge of classical computer programming, the difference between non-real-time and real-time systems must be especially emphasised in a communication protocols course.

Modern protocols tend to by very complex, so it is quite difficult to specify them in a simple and unambiguous way. Traditionally, specifications were written in a language that was difficult to read and even more difficult to understand (called "bureaucratspeaklanguage" by prof. Tanenbaum [1]). Nowadays, formal specification languages are becoming more and more popular.

SPECIFICATION AND DESCRIPTION LANGUAGE

By specifying a communication system in an appropriate formal language, three basic goals can be achieved at the same time [2].

- The system model is conceived in a precise and rigorous way.
- The system specification is simple, unambiguous and easily understandable.
- The system specification is machine readable and interpretable.

From the didactic point of view, there is an additional, but very important fact.

- By using formalisms, students get accustomed to reason in a rigorous way that closely fits models of designed/analysed systems that are imposed by the language.

Therefore, we consider teaching formal specification languages to be didactically an essential component of a communication protocols course. Actually, a teacher may impose the way students think about communication systems by choosing an appropriate language.

Since long ago, physicists and engineers have been using mathematical notation to express their reasonings and results. Later on, the way of reasoning has been changing, so more and more new formalisms have been needed, being able to describe systems from different veiwpoints and at different levels of abstraction.

Nowadays, three languages are considered especially suitable for comunication protocol specification, namely LOTOS [3], ESTELLE [4] and SDL [5]. All of them have been standardised, LOTOS and ESTELLE by ISO, and SDL by ITU. While the first one is algebraically oriented, the other two specify the functionality in a more procedural manner. Furthermore, SDL offers the graphical syntax in addition to the textual syntax, which makes it especially suitable for educational purposes (graphic functionality specifications are quite similar to flowcharts frequently used to describe computer programs and algorithms). SDL is used as the specification language by International Telecommunication Union (ITU) and European Telecommunication Standards Institute (ETSI) which makes it very important also from the practical viewpoint.

SDL (Specification and Description Language) was developed and standardised by International Telecommunication Union (ITU) for the specification of telecommunication systems and their components, although it is suitable for the specification of any real-time systems [6]. Using SDL, both structure and functionality of systems can be specified.

SPECIFYING TELECOMMUNICATION SYSTEMS WITH SDL

Specification of Communication System Structure

The structure of a system is described by means of blocks and channels with which blocks are interconnected. Information is transferred via channels in form of signals. A usual structure when specifying a communication protocol is shown in Fig. 2, where the `ServiceProvider` is either the whole communication system or only one of its layers. In the first case, the users are real communication system users, while in the second case, the users represent the upper layer of the communication system. Signals `req`, `con`, `ind` and `res` model the primitives for inter-layer interaction.

In the light of the hierarchical design paradigm, supported by SDL, the block `ServiceProvider` can be further refined as shown in Fig. 3. Here `(pdu)` specifies the set of protocol messages and blocks `EntityA` and `EntityB` model protocol entities. The functionality of an SDL channel is very simple (no losses, no delays), so a special block (like the block `Ch` in Fig. 3) can be used to model a more sophisticated channel.

It is necessary to emphasise that the possibility of SDL to specify structure hierarchically is very important from methodological viewpoint, as it reflects the principles of abstraction and information hiding that are heavily used in modern communication and other technical systems.

Specification of Communication System Functionality

For any block, either its substructure or its functionality must be specified. In SDL, a system functionality is specified in terms of processes. The basic theoretical model of process functionality is extended state machine, i.e. state machine which not only can stay in one of its states and transit from one state to another, but also can store values in its variables which substantially reduces the number of states and thence the process complexity. In addition, an SDL process can have and manage its own timers. All input stimuli (input signals and timer expirations) enter the process through an implicit input queue. In fact, this is the model we have seen already in Fig. 1. Hence, an SDL process is very suitable to model a protocol entity. Therefore, if SDL is chosen as the formal language used in a communication protocols course, students are somehow forced to think about protocol entities in terms of the model shown in Fig. 1.

SDL signals are supposed to be born when they are output by a process and to die when consumed by another process. Furthermore, they can bring several values of different types with them. In both of these properties, they are similar to protocol messages which can therefore be modeled by them.

Signals that are transferred between processes and values that are brought by signals and stored/processed by processes characterise very well the information which is transferred, stored and processed by communication systems and can have either dynamic or static nature. On the other hand, processes that either can make decisions based on static conditions ("if") or can wait for input events to occur ("when") are excellent representatives of systems that operate in real time. So SDL can serve as an excellent vehicle to explain the essential operation of discrete-event systems to students.

COMMUNICATION PROTOCOLS COURSE

In previous sections, the role of formal languages in teaching communication systems and communication protocols was exposed. The presentation and use of a formal language were shown to be essential elements of a communication protocols course. In this section, the presentation and use of the SDL language and especially their pedagogic purpose in a communication protocols course held by the author at the University of Ljubljana, Faculty of Electrical Engineering, will be presented with more detail.

Lectures

During the lectures, the basic elements of communication systems, such as protocol entities and channels (including the protocol entity model shown in Fig. 1) and the usual communication system structure (protocol stack) are first presented. Then the need for formal languages and their role in communication system structure and functionality specification are explained.

The syntax and semantics of the SDL language are taught in more detail. Graphic syntax is used. The use of SDL for protocol stack structure and protocol entity behaviour specification is explained. The similarity between the models of protocol entity and SDL process is also emphasised. SDL constructs used to specify behaviour in real time are taught with special care to let students become aware of the peculiarities of such systems.

In the continuation of the course, different methods and algorithms used by protocols are taught. All examples are specified in SDL. In this way, students are becoming accustomed to SDL and, more generally, to use a formal language as a language of communication between engineers. Above all, they are becoming used to think about engineering problems more rigorously and more precisely. Consequently, formalism is becoming more and more the way of their thinking.

Lab work

In the lab, the set of tools GEODE [7] for the SDL design/analysis, that consists of an editor, syntax checker, simulator and compiler into C code, is used.

At first, the teacher explains how to use the editor and the simulator. Then, a very simple communication system is specified and simulated. However, the system is intentionally oversimplified (e. g., the timer is omitted which results in a deadlock). A channel with delay and losses is also modelled. This system is simulated step by step. As its defficiencies are discovered, it is upgraded to operate better. In this way, some logic errors in protocol can easily be discovered, explained and corrected. Furthermore, the situation that arises when a user generates more traffic than can be handled by the communication system can also be clearly seen and understood.

Home Work

Now, it's the students' turn to cary out a somehow more complicated task! They must specify a complete communication system (such as is shown in Fig. 2 and 3) and a complete protocol, including connection setup, information transfer and connection release phases, in SDL. Of course, the system and the protocol are very simple (it is about the Alternating Bit Protocol [8]), but have all the essential characteristics of communication systems. As SDL is only suitable to specify the functionalities of protocol entities (and not those of users and channels), students are given textual specification of the protocol (i.e. protocol entities' functionalities) and must transform them into formal specification. This is the best way for a student to come to know very well the protocol and its mechanisms, to experience the defficiencies of textual specifications and the advantages of formal specifications, as already listed in this paper, and, most importantly, to reason about the operation of real-time systems in terms of the model shown in Fig 1 and especially in terms of finite state machines. Students are intentionally given a specification that is incomplete (the specification of connection release is missing), so they are forced to figure out by themselves, how to close the connection.

The GEODE software is very expensive, so licences for using it are scarce. Therefore, it is not reasonable to let students play with the software, before they understand very well how to write the protocol specification in SDL. Hence, they must first compose their specifications on paper and discuss them with the teacher. Of course, the main reason for this is not only the lack of licences, but especially our desire to teach them to work with one's head, instead of just playing with a tool!

Only after the paperwork has been approved by the teacher (which is normally not at the first reading!) and, of course, if there's no lack of licences and time, students can be let to enter their specifications into the computer in simulate them, to find out syntactic and semantic errors.

In our case, students only do the paperwork. The reason for this is above all the lack of licences for the simulator. Although ckecking the correctness of a specification with each student separately is a substantial burden for the teacher, this work is deemed very important from the didactic viewpoint, as the teacher can easily understand what is really not understood by the student, and can therefore help them very effectively.

It is especially important that students understand the semantics of SDL, as they reflect both the basic model of a protocol entity shown in Fig. 1, as well as the basic functional model which is in fact the finite state machine. Hence, the paperwork is considered by far the most didactically important in the frame of a course giving the basics of communication protocols. To let students work with the editor and not with the simulator has not much sense, especially not from the viewpoint of protocol teaching. Therefore, let students use both editor and simulator in addition to the paperwork (to find out errors themselves), or do only the latter (and be helped by an instructor)!

At last, let us ask, if students can just copy their specifications one from another, as all of them have the same projectwork to do. Of course they can! However the teacher, when discussing with a student their specifications, can easily figure out whether a student understands the subject and hence has done the project by themselves or has just copied it. Additionally, students know very well that they will have to write another SDL specification at the exam!

CONCLUSIONS

The author of this paper has been teaching a basic communication protocols course at the University of Ljubljana, Faculty of Electrical Engineering, for a number of years. In spite of a small number of hours devoted to the course (30 hours of lectures and 30 hours of lab work), four hours of lectures and four hours of lab work are devoted to SDL. In addition, students have to specify the Alternating Bit Protocol (as indicated already in this paper) at home. Due to the lack of licences for the simulator, their work is checked by the teacher on paper only. This fact allows the teacher to tolerate minor syntactical errors which are not considered important. Once more: we want to teach the semantics of real-time systems based on the extended finite state machine model, and not the syntax of just another language! The homework must be done by every student, as this is a necessary condition to apply for exam.

The GEODE system offers more tools than have been mentioned here, but they do not fit into a basic protocols course.

When designing communication protocols, more formal languages, some of them even compatible with SDL, according to International Telecommunication Union, can be used, such as MSC (Message Sequence Charts) or ASN.1 (Abstract Syntax Notation One). However, in a course consisting of 30 hours only, it is not possible to cover all that material.

Before enrolling to the protocols course, students already learn some basics of programming in C/C++. Nevertheless, in the protocols course they first encounter some specifics of real-time systems. Evidently, those SDL constructs that are characteristic of real-time operation, such as waiting for an input event or timer management, seem to be most troublesome for them, especially for those that have already done some sequential programming. It is also difficult for them to understand the difference between values and signals. However, these are essential concepts of communication systems, and one of the most important tasks of a protocols teacher is to make these concepts clear!

Because our main purpose is to teach some basic principles of system operation in real time, the pedagogic principles described in this paper should not be applicable only for the education of telecommunications engineers, but for real-time systems engineers in general. Of course, for other branches of engineering, other languages can be more suitable.

ACKNOWLEDGEMENT

This work was partly supported by the Ministry for Education, Science and Sport of the Republic of Slovenia as a part of the project P2-0246 (Algorithms and Optimisations in Telecommunications). We also thank the company Iskratel d.o.o., Kranj, Slovenia, for allowing us to use their licences for the GEODE system in the laboratory work.

REFERENCES

[1] Tanenbaum, A., S., *Computer Networks, Third Ed.* , Prentice-Hall, Upper Saddle River: 1996.

[2] Lai, R., Jirachiefpattana, A., *Communication Protocol Specification and Verification* , Kluwer Academic Publisher, Boston: 1998

[3] ISO, "Information processing Systems - Open Systems Interconnection - LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour", *ISO 8807* , International Organisation for Standardisation, 1989

[4] ISO, "Information processing Systems - Open Systems Interconnection - ESTELLE (Formal Description Technique Based on an Extended State Transition Model)", *ISO 9074* , International Organisation for Standardisation, 1989

[5] ITU, "CCITT Specification and Description Language (SDL)", *Recommendation ITU-T Z.100* , International Telecommunication Union, 1993

[6] Bræk, R. Haugen, O., *Engineering Real Time Systems* , Prentice Hall, New York: 1993

[7] Encontre, V., "GEODE: an industrial environment for designing real time distributed systems in SDL", *Proc. Fourth SDL Forum* , North-Holland, Amsterdam: 1989, pp. 105-115

[8] Bartlett, K., A., Scantelbury, R., A., Wilkinson, P., T., "A note on reliable full-duplex transmission over half-duplex links", *Communications of the ACM*, Vol. 12, No. 5, 1969, pp. 260-261

FIGURE. 1
MODEL OF A PROTOCOL ENTITY.

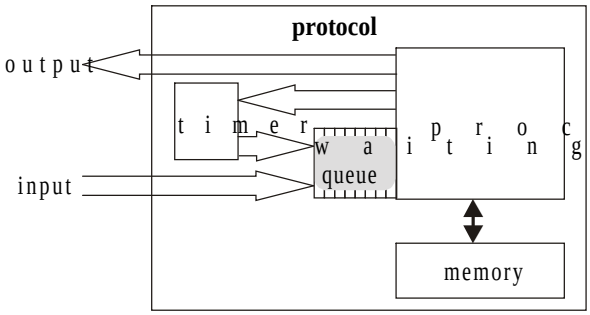

FIGURE. 2
STRUCTURE OF A COMMUNICATION SYSTEM.

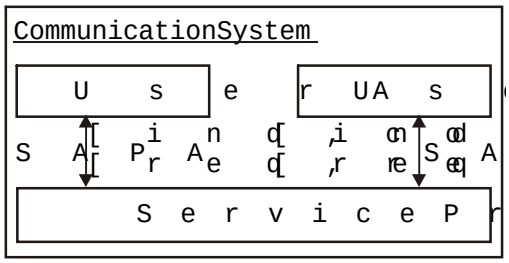

FIGURE. 3
STRUCTURE OF A SERVICE PROVIDER.