

A Graphical Environment for AUV Mission Programming and Verification *

N.A.Anisimov¹, A.A.Kovalenko¹, G.V.Tarasov¹,
A.V.Inzartsev², A.Ph.Scherbatyuk²

¹ Institute for Automation and Control Processes,
Far East Branch of the Russian Academy of Sciences
5 Radio Str., Vladivostok, 690041, Russia

² Institute for Marine Technology Problems,
Far East Branch of the Russian Academy of Sciences
5a Sukhanov Str., Vladivostok, 690600, Russia

Abstract

One approach for creation of a graphical environment for AUV mission programming and verification is described in the paper. It is based on a Petri net theory. The results of the AUV mission programming and verification using the graphical environment are supplemented.

1 Introduction

The mission programming of an autonomous underwater vehicle (AUV) should be acceptable for users with different knowledge of inside AUV structure. It should be provided possibility to use high level functions understandable for users concerned with a concrete application for more effective tasks solution.

Other important problem consists in programmed mission verification. It means that before launching the defined mission should be verified for logical aspects of correspondence of the programming to the desired goals (absence of deadlocks, conformity of the results to various scripts, ability of safety property in any fatal exception), temporal characteristics (absence of temporal deadlocks, values exceeding the programming bounds) at el.

One way for solving of the mentioned above problems consists in use of a languages with a well defined operational semantics. As examples it can be mentioned PROLOG [6] and ESTEREL [4]. But use of such languages demands high enough programming competence and may be complicated for end-users.

Designing of a development environment, which provides mission loading, verification and simulation tools along with a graphical human-machine interface is other way for AUV programming. One approach for graphical development environment creation is based on use of Petri nets theory [10, 8, 1]. The Petri nets theory was designed for the representation and analysis of parallel and distributed systems.

*Published in: Proceedings of the 10th International Symposium on Unmanned Untethered Submersible Technology, pp.394-405 (New Hampshire, USA, September 7-10, 1997).

Such approach was described in a paper [5]. Development environment ORCCAD firstly intended for controllers designing is used here for mission creation in terms of Petri nets. Common approach for description of AUV control system operation and mission creation in Petri nets terms is proposed in [9]. The methodology described in the paper is based on a key concept of Vehicle Primitive, which is a parameterized specification of an elementary operation performed by AUV.

At the same time, it is recognized that Petri nets are not suitable for straightforward application in practice. More convenient and user-friendly interface is needed as well as corresponding computer-aided environment. In this paper we suggest an approach in which AUV mission design techniques is strongly based on Petri net formalism. On the basis of Petri net model we are developing a graphical language for mission specification. The graphical language syntax is similar to Specification and Description Language (SDL) [11].

The organization of the paper is as follows. The section 2 discusses a method for mission program creation used for AUV of MT-series. In the section 3 we introduce some conceptual basis upon which the proposed techniques is relied. In particular, we consider both general and formal models of AUV mission and the graphical language for mission specification. A current version of the graphical environment for AUV mission programming and verification is described in the section 4. The results of the AUV mission programming and verification are supplemented.

The paper is a revised and extended version of the previous paper [3].

2 AUV Mission Programming Systems

The AUV control system architecture is subdivided usually into some levels. Three levels control system architecture is used in the AUV of MT series designed in the IMTP FEB Russian Academy of Sciences [7]. According to the conventional terminology [6, 9] it consists of organization (or strategic), coordination (or tactical) and execution levels. The execution level is connected with definite AUV body and serves for on board units and devices parameters and modes control. The organization level is responsible for AUV mission fulfilling. The coordination level is an interface between “asynchronous” organization level and synchronized execution level. It converts the mission commands flow from the coordination level into instructions for the execution level and translates the data and results of operations from the execution level for the coordination level.

2.1 A Language Based AUV Mission Programming

The language based approach was used initially for mission programming of MT series AUV. An AUV programming language is a subarray of C language. The AUV mission program consists of AUV motion and on board units and devices operation subroutines. The AUV motion subroutine defines the trajectory shape. The on board units and devices operation subroutines are built on base of motion subroutine and describe AUV operation during the trajectory realization.

The AUV programming language provides a set of embedded subroutines and system processes that can be subdivided into three groups. A first group defines the shape of the AUV trajectory. A second group includes a system processes for on board units and devices operation organization. At last a third group is intended for sensors data access and serves for back loop organization during mission fulfilling.

This AUV programming language allows to create effective AUV mission programs. Also it is

possible to use a commonly used trajectories and procedures library. From other side such programming language does not provide any mechanisms for mission verification and it is supposed high enough programmer qualification.

The AUV mission program intended for operation on board of AUV is prepared on the supported ship computer system. For verification of the designed mission program before it's transmitting into AUV the designed mission program simulation is organized. The simulation program includes AUV, control system and surrounding environment models. The AUV model serves for AUV motion calculation. The simulated trajectory is displayed on system monitor. The control system model is used for simulation of the controllers operation and surrounding environment model is intended for sensors operation simulation.

Correspondence of the programmed and desired missions is estimated on base of mentioned above simulation procedure. In case of need to improve the programmed mission it can be easily changed and newly simulated.

It can be pointed that the described system for mission program simulation is effective just for simple enough linear mission program. But presence of many program branches depending on surrounding environment conditions and AUV states makes the simulation process more complicated and less effective.

2.2 A Graphical Environment Approach for AUV Mission Programming

As mentioned above the important problem is to design the development environment for AUV mission programming, which allows for end users with different competence to create, verify and load mission program into AUV. The graphical environment based on PN³-Tool system [1] was created for solving of this problem. The graphical environment provides the AUV mission program designing, verification and than converting it into AUV language program described in section 2.1. The AUV language program is ready for translation and loading into the AUV on board computer.

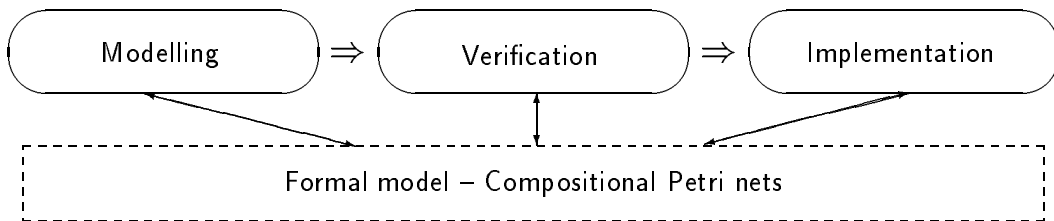


Figure 1: Schema of AUV mission design.

The mission program creation process is subdivided into three steps (Fig.1):

- mission generation - specifications producing in the graphical environment by means of the graphical language;
- verification and simulation - produced specifications analysis for logical and temporal correctness;
- realization - converting of the designed specifications into the AUV language mission program, which is ready for execution on board of AUV.

A graphical environment approach for AUV mission creation is described in the next two sections.

3 Conceptual Models of AUV Operation

3.1 Reference Model of AUV Operation

The most general and abstract model of AUV operation served as Reference Model is presented in Fig.2. The AUV model consists of one *mission object (MO)* and several *device objects (DO)* which are controlled by MO while executing of the mission. During the mission execution the MO communicates with the device objects via sending and receiving of messages. Two types of communication primitives are defined: *request/reply (RR)* and *event (E)* primitives.

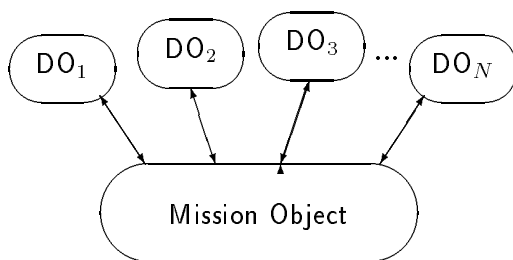


Figure 2: Reference Model of AUV operation.

Request/reply communication is executed by the following way. First, MO sends a request message to some DO for invoking of needed function. On receiving this message, DO invokes this function and returns a reply message with the results. The MO receives the reply and can continue the mission execution. It should be noted that during this interval the MO is blocked and can not do any useful job. Request and reply messages can bring some parameters. One request can cause different replies depending on the state of DO. For example, MO sent a request “turn on the photo camera” can expect one of two replies “the photo camera is turned on successfully” or “the photo camera is not ready”.

The event primitive is initialized by one of device objects. If some event that is relevant to mission execution occurs in any device object, then this DO sends a event message into MO. For example, DO corresponding to photo camera can send the message event “my film is ran out”. Naturally, the MO must be ready for processing such unsolicited events.

In order to distinguish different device objects, each of them is being identified by a unique address. Therefore each message in communication like request, reply and event brings this address inside it as a parameter.

3.2 Petri Nets Representation of the Reference Model

According to the Reference Model, the AUV mission can be thought of as an object that operates by communication with device objects. In other words, the AUV model is a set of concurrent and communicating objects. However, for design and verification purposes we need in more formal representation of AUV mission. In this section we present precise mathematical model of AUV mission that is a formal refinement of its Reference Model. This model is based on compositional approach to Petri nets. Now we shortly and informally recall some basic notions

from compositional Petri nets. More information on this subject can be found in the literature, see [10, 8, 2].

Petri net is usually defined as a bipartite graph with two types of nodes - *places* and *transitions* represented as circles and boxes respectively. *Marking* of the Petri net is a distribution of *tokens* over places represented graphically by black dots inside of the circles. The transition is said to be *enabled* if each its input place has at least one token. The enabled transition may fire. Firing of the transition removes one token from each input place and adds one token to each output place. Petri net with initial marking can dynamically operate by firing of transitions.

Petri net entity is a Petri net equipped with a set of *access points*. Each access point has a name and a mapping that assigns *labels* to transitions. Graphically, a Petri net entity is represented as Petri net where each transition is labeled by an expression of the form *ap:label* where *ap* is a name of access point and *label* is a label of the transition. This net is placed inside a box with emerging arcs, each representing access point. An example of Petri net entity is represented on Fig.3. Two entities can be united by the operation of entity composition via access points. In short, composition of two entities E_1 and E_2 via access points α and β results in a new entity by union of corresponding Petri nets and merging transitions visible from α and β by the same labels.

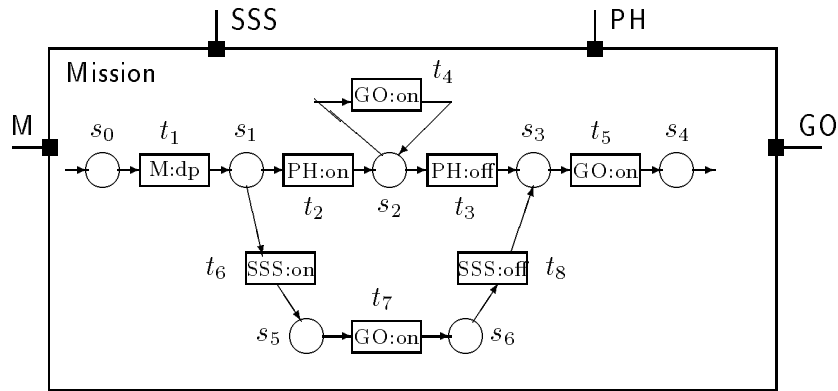


Figure 3: Example of AUV mission based on the Petri net entity model.

Additionally the Petri net entities are loaded by operations with variables as it does for high level Petri nets [8]. Particularly a set of definite types variables are defined for entities. Each transition has a predicate and a procedure defined for that variables. It is considered, that transition is enabled when it is enabled in ordinary sense and more over its predicate is fulfilled. When transaction fires its procedure is invoked that can change variables meanings.

Now the Reference Model depicted on Fig.2 can be shown as Petri net entities combination, added by composition rule through access points. Particularly the AUV mission is shown as Petri net entity with access points that intended for interaction with definite devices.

The example on Fig.3 describes a simple mission with two alternative ways. The mission starts with depth measurement (transition t_1 with access point $M : dp$). In a case if depth is less then 5.0 meters the mission follows the upper flow graph. According to highest graph the AUV must turn on the photo-camera with the period of shooting equaled to 10 seconds (transition t_2 with access point $PH : on$). Then the vehicle is moving in a right rhombus way with edge equaled to 4 minutes (the cycle with transition t_4 with access point $GO : on$). This way is ended by turning off the photo-camera (transition t_3 with access point $PH : off$). Another alternative way of the

whole mission's graph will be used in case if depth is equaled or more then 5.0 meters. In that case the AUV will move to the 60 degrees direction while writing the SSS information (the flow of transitions t_6, t_7, t_8). The last mission action is to return to the initial space point (transition t_5 with access point $GO : on$).

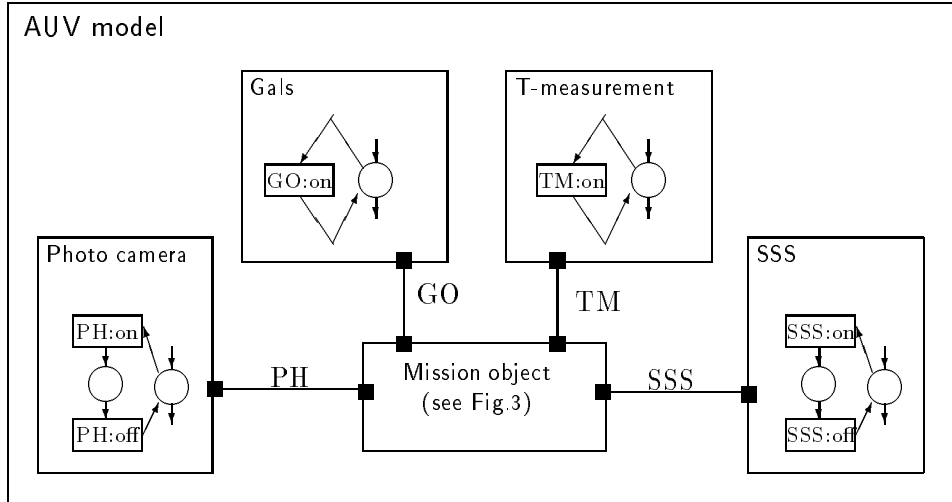


Figure 4: AUV operation model integrated with units models.

The model based on Petri net entity can be built for each AUV unit. The model will have one access point for interaction with mission object. The AUV operation model is a combination of all units models, based on composition rules. The AUV operation model using the units models is illustrated on Fig.4.

It should be pointed, that the resultant Petri net does not have the access points. In other words it is the ordinary Petri net. The AUV mission correctness analysis leads to the Petri net received verification. Initially the reachability tree is built [10]. Then the constructed tree is subjected for analysis the results of which comprise the information like boundness, safeness, liveness, etc. The absence of deadlocks and leavelocks proves that mission will be fulfilled.

3.3 Graphical Language Representation of the AUV Operation

As it has been pointed out in previous section, the AUV mission can be completely specified in terms of Petri nets. It means that the process of mission specification can be achieved by constructing of corresponding Petri net entity. However, despite the fact that Petri nets possess of good graphical interface, its utilization requires special qualification. Therefore we need in more convenient graphical notation that can be easily used by wide range of practitioners who are unfamiliar with Petri nets. For this purpose we are designing a language with SDL-like graphical syntax [11] while its semantics is strongly based on Petri net entity formalism.

In Fig.5 only two basic operators of the graphical language are depicted: request/reply and event operators. As it can be observed from the picture, each operator has an icon corresponding to operation with device object and a name of this operation. The request/reply operator may have fields corresponding to different replies (e.g. **ok**, **err**). Besides, the language has some standard control operators like state, begin, and etc. The mission programmed on this graphical language can be drawn as graph with operators as nodes.

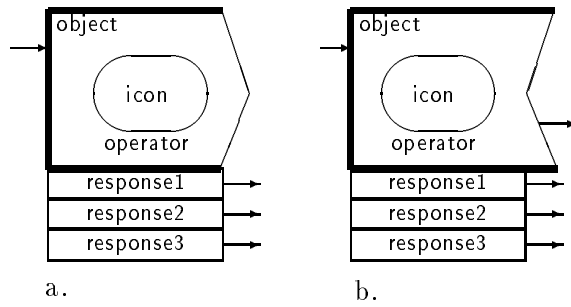


Figure 5: Graphical language operators.

4 Graphical Environment for AUV Mission Creation

A graphical environment structure for mission development is presented on Fig.6. It consists of three subsystems: mission editor, verification and implementation subsystems. The special subsystem for user interaction is based on the standard Windows interface. These subsystems are intended for:

Figure 6: The graphical environment structure.

1. *mission editor* allows to create visual programs corresponding to the needed tasks. The program itself is generated by means of special graphical language, which is mentioned in the previous section. The editor has the capabilities for creating and modifying the programs in Petri net representation. Also, mission editor is intended for the AUV devices specifications creation and modification. Those specifications are needed for verification subsystem.
2. *verification subsystem* was developed for analyzing of the ready specifications. In the existing version of the environment this subsystem allows to check the specifications for logical correctness. In terms of AUV it means checking of the devices correct usage, specification completeness (the absence of unspecified events), absence of excess specifications, etc. This subsystem uses the mission program with device specifications in order to create the model of its interaction. Then the model is analyzed by means of Petri nets methods.
3. *implementation subsystem* processes the final specifications and produces the program in AUV programming language. Further, these program can be translated and loaded into the AUV computer system.
4. *user interaction subsystem* controls all other subsystems via standard elements of Windows GUI: menus, dialog boxes, toolbars, etc.

4.1 Graphical Editors for Mission Specification

The graphical environment suggests two modes for edition of AUV missions. One editor intended for end users uses graphical language presented in the Section 3.2. The other editor is used by qualified system programmers and allows to upgrade and correct elements of user editor in terms of compositional Petri nets.

4.1.1 User Editor

The User Editor is a multi-window graphical editor that allows to design the AUV mission program in terms of graphical language pictograms. The editor has the main toolbox with buttons, where each button represents a definite device object. Each button in its turn is associated with a set of device functions pictograms. The designer can select needed pictogram and drag it onto current page. There are means for connecting of pictograms by oriented arks. There are also means for parameterization of operators like setting possible replies for RR operators, conditions and actions on variables. Fig.9 illustrates the example of AUV mission created by means of User Editor in its graphical environment.

Figure 7: User Editor graphical environment.

After the mission creation process finishing, user must verify mission specifications for logical properties. This may be done by means of analyzer. Indeed, pictogram specifications are transferred into the Petri net representation, and the resultant net is subjected for verification. When verification is successfully fulfilled the implementation subsystem produces the program in AUV programming language. These program can be translated and loaded into the AUV computer system.

4.1.2 Petri Net Editor

This subsystem is intended for use by designers who have to produce the models of AUV devices. Moreover, a designer interested in a formal underground of pictogram's specification may use the Petri Net (PN-)Editor. The process of mission construction in PN-Editor comprises Petri net drawing with setting up the properties for elements (places and transitions) of the net. The

resulted specification describes the control flow of mission, while the properties of elements allow working with data variables. In general, PN-Editor supports the following capabilities:

- Multi-windows specification construction. Each window corresponds to some procedure, which can be called within the transition of another procedure. The special dialog box offers the mechanism for controlling the set of windows (creation, deletion, naming/renaming, switching, picture scaling, etc.)
- Petri net drawing: putting/removing/relocating of nets places and transitions, connecting of the elements by means of oriented arcs. Setting up of the properties for elements. Place can be mark as normal (internal), head or tail. The information about the type of a place is needed for creation of a code-sequential program. Transition properties can be assigned via special dialog box has the access points, condition and action fields. Access points are used for verification purposes in order to construct a model of the mission via parallel composition of the mission object with all device objects. Access point can be entered either manually or by choosing from the dialog box showed at right bottom corner of Fig.8. Condition field specifies the possibility of transition firing, while the action field declares the action which performs if transition firing occurs.

Figure 8: PN-Editor graphical environment.

- Petri net construction: means for algebraic operations, like sequential (head and tail places of different nets merging) and parallel (transitions with similar access points projections merging) composition, choice composition(merging of head places and merging of tail places for two nets), disruption (a complex operation which transfers a control over flow from one net to another) and iteration (merging of head and tail places for the same net). All these operations are performed using a friendly dialog box which offers for user to set up the type and operands of operation and the output window.
- Architecture modeling subsystem allows to manipulate with windows procedures as a "black" boxes with access points. This editor is useful for the specification of mission

object interaction with different device objects. On the level of the architecture modeling subsystem the user operates with boxes (each represents Petri net procedure) and its interconnections.

- Animation subsystem lets to “run” Petri nets that means the visualization of tokens jumping over net process. There are two different modes of running nets: interleaving (only one of possible transitions fires) and step’s (several excited transitions can fire) transition firing. The user can control the process of firing by switching it from manual to automatic with specified delays. This subsystem can be used for visual investigation of dynamic Petri nets properties.

4.2 Analyzer

The subsystem of specification verification for logical correctness is based on reachability tree construction for Petri net that describes the model of the mission. This mission model is received as a product of parallel composition between all objects specifications in terms of Petri net (including all devices and mission object). The complexity of produced tree depends only on the availability of computer resources. Then the constructed tree is subjected for analysis the results of which comprise the information like boundness, safeness, liveness, etc. Using these basic properties the subsystem can check the following reachability of the mission. The absence of deadlocks and leavelocks (infinite cycles without exit) (except final states) guarantees that mission will be fulfilled.

Figure 9: The results of mission program verification.

The user can view the results of the analysis at special dialog box, shown on Fig.9. If checkbox “Mission is reachable from all states” is absent in check state, then the below listboxs explain the source of the problem. For convenience, the user can see the program path which lead to this error situation.

Figure 10: The mission in terms of the AUV programming language.

4.3 Implementation Subsystem

The final aim of mission construction is to receive the executable mission program for AUV. So, the implementation system (implementator) parses mission object specification in terms of Petri nets and produces the program in the AUV programming language. This program can be translated by means of any C translators and then be loaded in the AUV computer system. The rules of mission parsing algorithms are as follows:

- Every place produces a label in the resultant program. The code that follows this label comprises the set of conditions with actions and “goto” statements - each corresponds to the condition and action field of outgoing arc.
- The starting execution point corresponds to the head places of the net.
- There may be several tail places – each must have the “exit” statement.

The implementator write the mission program in the AUV programming language into the file with the same name as a mission procedure name. The user can subjected this file for editing in order to add more variables or change some statements. The resultant mission program in the AUV programming language is shown on Fig.10.

5 Conclusion

A current version of the graphical environment for AUV mission programming and verification is described in the paper. The results of the AUV mission programming and verification using the graphical environment are supplemented. Now we are planing a next version of the graphical environment for AUV mission programming and verification including extended version of graphical language and verification subsystem.

6 Acknowledgments

The work performed by the first three authors was supported by the Russian Fund for Basic Research (Project No. 96-01-001773).

References

- [1] N.A.Anisimov, A.A.Kovalenko, P.F.Postupalsky, Compositional Petri Net Environment. In: Proc. 1994 IEEE Symposium on Emerging Technologies & Factory Automation – Novel Disciplines for the Next Century (ETF A'94), November 6 - 10, 1994, Tokyo, Japan, IEEE Press, pp.420-427.
- [2] Anisimov N.A., Koutny M., On Compositionality and Petri Nets in Protocol Engineering. In: Protocol Specification, Testing and Verification, XV. Eds. P.Dembiński, M.Średniawa, Eds., Chapman & Hall, 1996, pp.71–86.
- [3] Anisimov N.A., Kovalenko A.A., Insartsev A.V., Scherbatyuk A.Ph. A graphical environment for AUV missin programming, in: Marine technology (Ed. Ageev M.D.), Iss.1, Vladivostok: Dalnauka, 1996, pp.6-20 (in Russian).
- [4] Berry G., Gonthier G. The Synchronous Programming Language ESTEREL: Design, Semantics, Implementation, Science of Computer Programming. Vol.19, No.2, pp.87-152, 1992.
- [5] Espiau B., Simon D., Kapellos K. Formal Verification of Missions and Tasks. Workshop: Undersea Robotics and Intelligent Control, Lisboa, Portugal March 2-3, 1995.
- [6] Healey A.J., Macro D.B., McGhee R.B., Brutzman D.P., Cristi R. A Tri-Level Hybrid Control System for the NPS PHOENIX Autonomous Underwater Vehicle. Workshop: Undersea Robotics and Intelligent Control, Lisboa, Portugal March 2-3, 1995.
- [7] Inzartsev A.V. Mission Planning and Execution for Inspecting AUV. OCEANS'95, October 9-12, 1995, San Diego, USA.
- [8] K. Jensen, G. Rozenberg (Eds.) High-level Petri Nets. Theory and Application, Springer-Verlag, 1991, 724 p.
- [9] Pascoal A., Silvestre C., Oliveira P., Fryxell D., Silva V. Undersea Robotics Research at IST: The AUV MARIUS Programme. Workshop: Undersea Robotics and Intelligent Control, Lisboa, Portugal March 2-3, 1995.
- [10] Peterson J.L. Petri Net Theory and the Modelling of Systems, Prentice-Hall Inc., 1981.
- [11] Recommendation ITU-T Z.100. CCITT Specification and Description Language (SDL), 1993.