Bluetooth implementation frameworks

HARMATNÉ MEDVE Anna
Department of Information Systems
University of Veszprém, Veszprém, Hungary
Institute of Information Technology and Electrical Engineering
H-8200 Veszprém, Egyetem u. 10.
HUNGARY

Abstract: - This paper will represent an SDL implementation of Bluetooth mobile system's protocol stack. This implementation is part of a research and development of validation protocol system's implementation. The final goal is to develop a method for protocol implementation frameworks using design patterns of Protocol Systems in SDL. This work represents common problems and solutions for specification, design and implementation of communication protocols, furthermore it also gives a prototype implementation framework for Bluetooth's usage models.

Key-Words: - Protocol,Bluetooth, Reverse Engineering, Design Pattern, SDL.

1 Introduction

The large size, currency and real-time nature of mobile communication systems represent a set of difficulties when generating a good specification. The software to be developed must be flexible and portable enough to be employed into different environments and different platforms for different customer segments. The usage of formal methods makes the quick implementation of complex systems possible.

The SDL [1,2,3] is a standard language for specification and description of communicating systems. SDL currently has a dual role as a specification and also an implementation language. Most of commercial tools [4] have more functionality, including automatic code generation, simulation and validation. These are the main advantages of system modelling by SDL and it opens new possibilities for formal analysis of design. SDL provides the mechanisms to copye with large number of communicating processes, and there are SDL tools that give necessary features so that they can be used as protocol implementation frameworks

By using the domain specific patterns it is possible to describe structures and interactions of the systems. The detected SDL patterns could help in a run-time analysis of SDL models.

SDL patterns have particular context while conventional design patterns are specified by an independent design language – a mostly natural language–based description pattern. The language of SDL patterns is the SDL itself, SDL patterns are defined in terms of SDL syntax.

Protocol systems offer a multitude variety of services on different networks with a number of service options. They can be described by the SDL patterns - presented in Section 2.1. It is becoming very valuable to understand, maintain and re-engineer SDL models. An object-oriented analysis method needs to be integrated into SDL, the SOMT method (SDL-oriented Object Modelling Technique) [7]. As the name indicates, the method is essentially the adaptation of OMT to the requirements given by the special application area of distributed, reactive, real-time systems together with the usage of SDL for the design.

The Problem Formulation includes three main parts: section 2.1 „Patterns for Protocol System Architecture and SDL patterns" contains background information about communication protocols and their patterns [5,6], and we point out how it is possible to create an implementation framework of protocols by SDL. The section 2.2. gives an introduction to the context of Bluetooth usage models. Section 2.3 is„The Bluetooth implementation framework". Section 3.1is an application of them by producing a Bluetooth SDL pattern for the context presented in section 2.2. Finally in the Conclusion we will relate to the possibilities offered by SDL-2000 to enrich actual patterns

+ **The lecture was made under research contract OTKA 29556.**

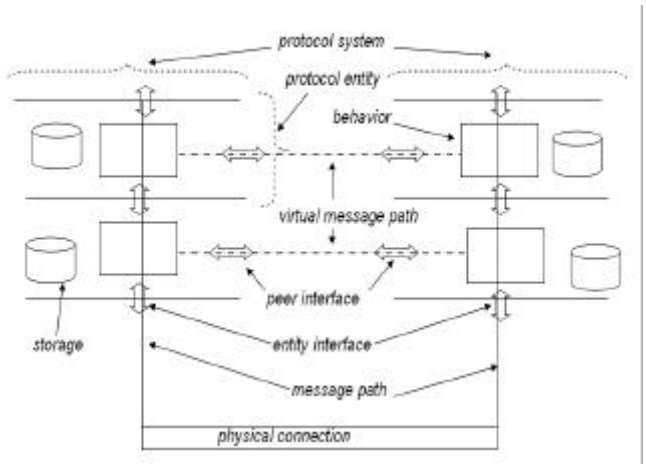
2 Problem Formulation

2.1 Patterns for Protocol System Architecture and SDL patterns

Protocol systems offer a multitude variety of services on different networks with a number of service options. The protocol systems are organized as series of subsystems, often called “protocol layers” or “protocol entities”. Conceptually different functions are separated into different layers and implemented separately.

Figure 1 describes two protocol systems communicating with each other via physical connection. Protocol entities communicate with each other by sending messages. Protocol stacks are connected by means of using psyhical connection, which represents the network. The entities in the same stack are connected to each other via message paths, they carry the messages within the system.

Protocol entities are connected to their peer entities using virtual message paths, the messages outside the system are sent on them. Peer communication is virtual as the messages sent to peer entities are actually sent using the interface provided by the lower protocol entity.



1. Figure *Protocol system elements*

The behaviour of a protocol is specified only in terms of protocol messages answering the question „ how protocol entities within a system communicate with each other”. Communication of an entity can be connection-oriented and/or connectionless. A connection-oriented communication consists of connection establishment phase, message excange phase and disconnection phase. A connectionless communication is a simple message exchange by Request-Response couple or only Request.

Research [6] on several existing protocol implementations contains the elements shown in figure 1. The same high-level model can be found in several protocol frameworks, including SDL [2].

The common general parts and relations in different protocols can be identified and described as design patterns [5]. In figure 2 we will represent three important patterns for protocol system architecture, which can be considered as architectural patterns and can be used in protocol-engineering.[6]

The *Protocol System* pattern - as shown in Figure 2 - specifies the components of a system, responsibilities and interconnections of components and the system environment. The *Protocol Entity* pattern represents one protocol layer, the *Protocol Behavior* pattern contains the active parts of a protocol entity.

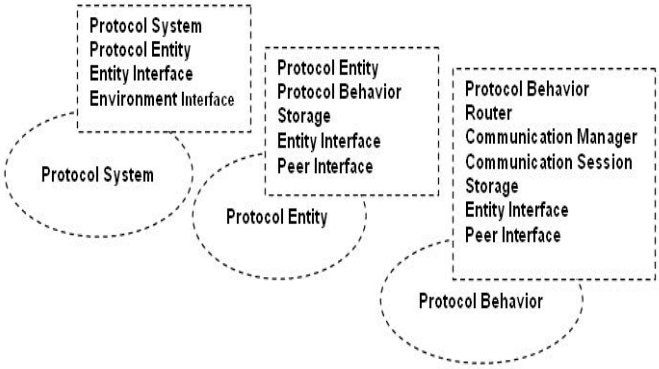


Fig 2: *Patterns for protocol system architecture*

The Protocol Architecture patterns components mapped in SDL its the following:

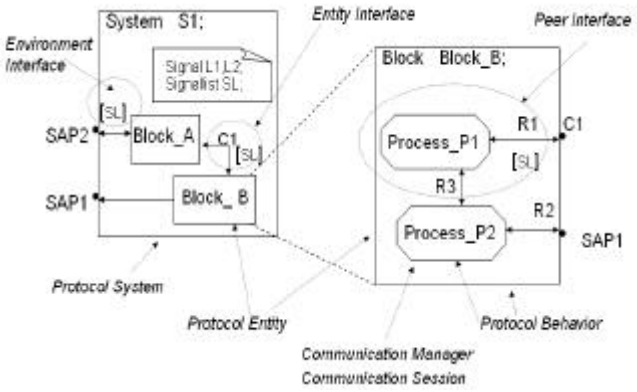


Fig 3: *Protocol Architecture pattern in SDL*

An SDL block type forms a *Protocol Entity pattern* : it represents a protocol layer or sublayer. Protocol Entity has to have communication accesswith other entities in the same system and with entities in peer systems. It manages possible multiple concurrent communication session, stores internal states and other information. Storage contains all information of a Protocol Entity. The behaviour of a block can be derived from the behaviour of its processes. The Peer Interface functionality is implemented in the process interaction

part by processes and signal routes. The Protocol Entity serves multiple requests and at the same time dinamically created and destroyed SDL process instances, too.
An SDL process type models a *Protocol Behaviour pattern.* The Protocol Behaviour element handles protocol functionality by the Communication Manager (creates, controls and closes sessions) the Communication Session (handles communication between peer entities) and Peer Interface elements of patterns (modeled by process instance and interactions between processes).
These protocol features always have to be implemented. The reusable components are defined as SDL types. The SDL package consists of libraries that are used for making reusable SDL components available in protocol systems.
In this passage I introduced my researches on correspondence between the elements of protocol system patterns and the elements of SDL. For us this research proves that if we improve protocol implementation with SDL development tools, it is not necessary to construct the whole description of protocol analysis neither in pattern language nor in UML for the requirements of re-use theory. We can construct SDL protocol patterns by means of applying the SOMT [7] method and Telelogic Tau^{TM} library modules in SDL development environment. SDL package may be used in the implementation of SDL frameworks. The packages are libraries that are used for making reusable SDL components available in different systems.

2.2 Bluetooth network usage models

Bluetooth is the name of a new short-range radio link technology developed by SIG (Bluetooth Special Interest Group), the standard is opened and the specification is downloadable from the SIG web site [8]. It helps to connect portable or fixed devices without cables. The Bluetooth radio module operates in the unlicensed 2.4 GHz ISM band using 79 or 23 channels with FHSS (Frequency Hop Spread Spectrum) scheme for avoiding interference from other signals in this band. The transmission works at 1 Mbps at a distance of 10 or 100 meters. A link can be ACL (asynchronous connection-less) for data transfer or SCO (Synchronous connection-oriented) for voice transfer. The standard supports both point-to-point and point-to-multipoint connections as well. The system follows master-slave pattern, but the master can collect more slaves (max. 7) to a piconet. A group of piconets in which the connections between different piconets are called scatternet. This network can mix heterogeneous applications, devices and usage models .

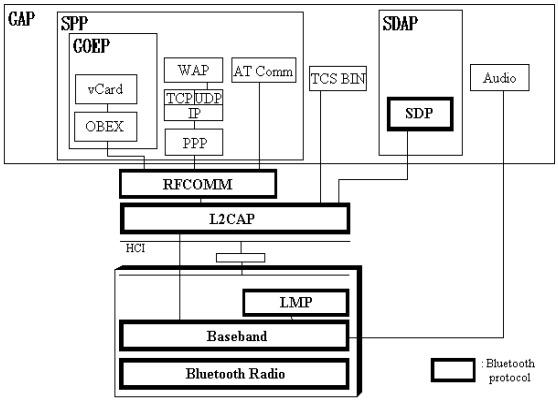


Fig 4: *The Bluetooth protocol stack and profiles*

The SIG has defined these protocols in the specification and determined some basic profiles for Bluetooth. A profile is (one or more) vertical slice in the protocol stack describing the mandatory protocols and parameter ranges for different user scenarios.
The used protocols are application-dependent, but the base Bluetooth protocols (Bluetooth Radio, Baseband, LMP, L2CAP, SDP) are used in every cases - except audio transfer.[4]
There are four general profiles determined by covering the common user scenarios. (Figure 4.) The Generic Access Profile (GAP) handles discovery and connection establishment between unconnected devices.
The second defined profile is the Service Discovery Application Profile (SDAP). It is responsible for searching for specific or general services in the range of the Bluetooth unit. SDAP re-uses parts of the GAP.
The Serial Port Profile (SPP) emulates serial ports on two devices and connects them with Bluetooth. It is used in the case of dial-up network, fax, headset or LAN access. This profile re-uses the pattern of GAP, too.
Finally the Generic Object Exchange Profile (GOEP) defines the protocols needed for applications using object exchange. This kind of profile can be File Transfer Profile, Object Push Profile or Synchronization Profile. GOEP uses GAP and SPP, so protocol engineers, who work out protocol stacks for object exchanging Bluetooth devices, can re-use GAP and SPP implementations.
In practice, for every usage model there is one or more adaptable profile. There are situations where the tasks are similar to each other, the used protocols are the same even in a different manner. In these cases it is practical to use one of the achievements of the object oriented protocol technology: reusing profiles.
The L2CAP protocol layer handles the various packages arriving from different applications and includes the protocol-multiplexing function the effect of which is the increasing number of use-cases of Bluetooth.

In the Bluetooth usage models there are several possible applications over L2CAP to communicate with each other. Such applications can be over the different LANs (wired or wireless [9]) based on IP protocol, over the object exchanging protocols based on OBEX, the telephone, fax or point-to-point modem just like the simple audio transfer. Thanks to L2CAP these applications can communicate with each other in every variation. The formal realization of these cases only needs to work out the different application scenes. If we want to use the same upper layer protocol to describe a part of a communicating situation, we can reuse the earlier predefined protocol's package.

2.3 The Bluetooth implementation framework

A framework is a reusable design of a complete system (or part of it) that is represented by a set of abstract classes and the way of their interaction [5]

The Bluetooth implementation framework is a set of SDL packages for protocol system patterns. In section 3 I will ilustrate the L2CAP protocol entity description.

SDL package containes the static and dynamic parts of protocol entity specification and their data descriptions. For pattern construction the data definitions are needed (ASN.1), because ASN.1 data-type definitions and inheritances are needed so that we can re-use the patterns by means of redefining SDL sorts.

For example in figure 5 one can see that protocol L2CAP executes multiplexing in all network models. The multiplexing function, the segmentation and reassembly (SAR) operation are implinks in the same way as the PDU's data type definition is in ASN.1 [11] using CHOICE type.

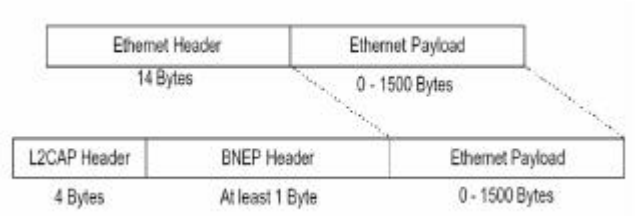


Fig 5: *BNEP with an Ethernet payload sent using L2CAP multiplexing and SAR*

3 Problem Solution

3.1 Presenting the Bluetooth L2CAP protocol analysis and formal description using L2CAP pattern

3.1.1 Protocol analysis, defining system requirements

The structure, process and division of communication take place on the basis of master-slave relationship initiated by the master of piko network. The following status occur during system operation on the grounds of specification:

- Closed: status of both master and slave position is possible, the connection is closed
- W4_L2CAP_Connect_Rsp: after having transmitted the master's respond sign to the slave referring to the question of connection setting, the master is waiting for the slave's answer
- W4_L2CA_Connect_Rsp: this status is only peculiar to slave, having introduced the connection setting request, it is waiting for its respond
- Config: this status can be taken up by both the master and the slave following a successful connection setting or during the communication intended to promote agreement on channel particulars
- Open: this status is capable of being taken up by both parties for the sake of successful communication flow following the determination of channels
- W4_L2CAP_Disconnect_Rsp: the master gets here after having sent its disconnection request to the slave and is waiting for the slave's respond
- W4_L2CA_Disconnect_Rsp: after having received the disconnection request from the master, the slave informs its higher layer about this fact and shifts to this status waiting for the respond of the layer situated above

The service primitives (request, indication, response and confirmation) are signalled by L2CA_, while the PDU-s are completed with P (protocol) as, can be see in figure 7.

During the analysis MSC's scenarios [10] are needed for solving the timing problem and for the differentiating illustration of possible application cases (figure 6).

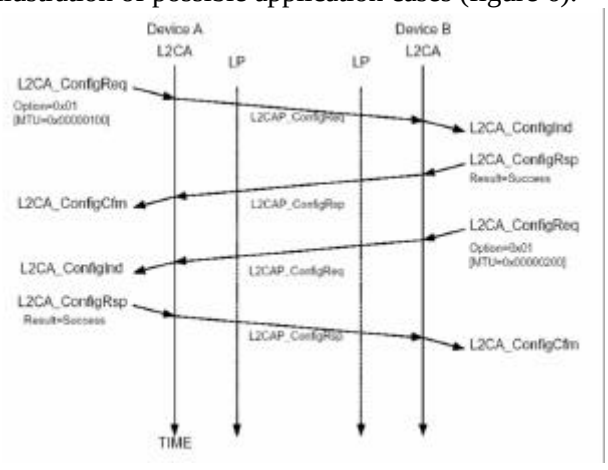


Fig 6: *In the basic configuration process the devices exchange Maximal Transmission Unit information*

With the help of all the status being distinguishable in the specification, service primitives and PDUs I drew a

state-flow graph (Figure 7) in which the operation of both the slave and the master are described at the same time.

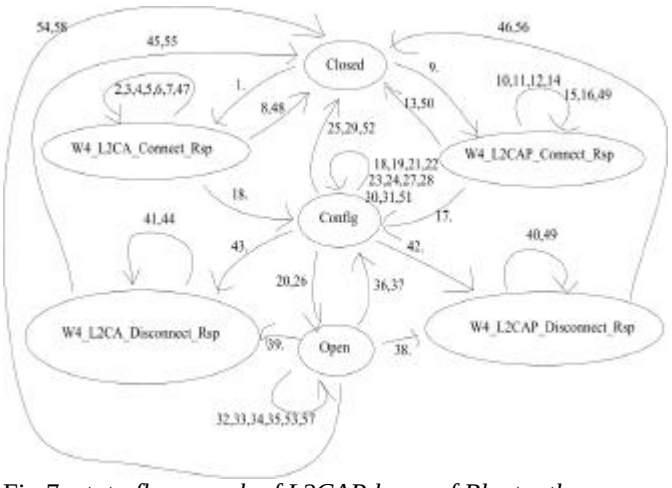


Fig 7: *state-flow graph of L2CAP layer of Bluetooth*

3.1.2 Preparation of formal description of protocol L2CAP

SDL description is structured hierarchically. On the highest level we can find the system-level description with the illustration of communicating parties, namely the L2CAP layer of the two Bluetooth objects, the signs used by them, and channels functioning on the basis of FIFO theory between objects carrying out information-exchange. The two parties L2CAP layer and the signals can be derived from the pattern described in the package (Figure 8).

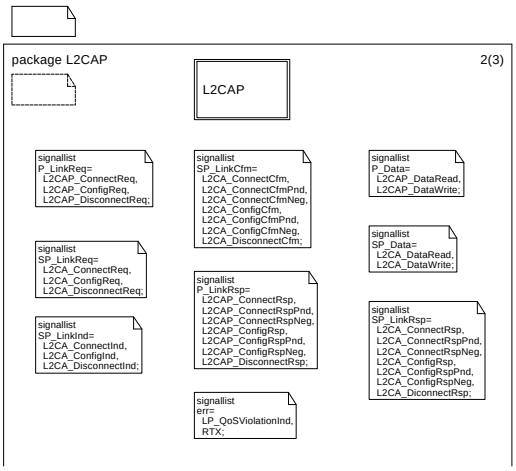


Fig 8: *A part of protocol L2CAP package*

Reusing that we can implement those elements only once (in the package patterns). In our case Figure 9 perfectly demonstrates the two system-level elements (master = L2CAP_Ini, slave = L2CAP_Resp), the PDUs used in communication between them, and the communication channels.

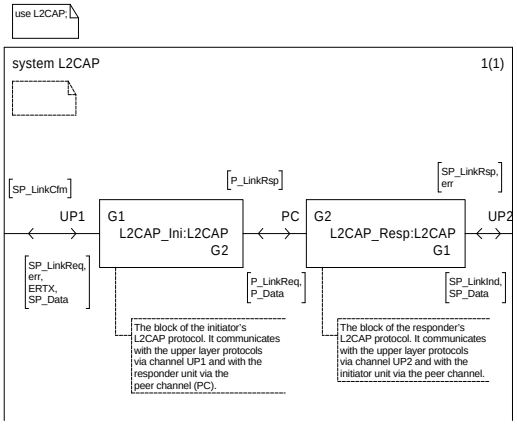


Fig 9: *A part of protocol L2CAP system-level description, which illustrates the members taking part in communication*

On the following hierarchy-level I will indicate the process interactions of block-level description breaking further down the L2CAP layers of master and slave. It also shows the inner development of the element, the included functional units (processes) and their communication channels, the signs used for information exchange. Let's take a look at the block diagram of L2CAP_Resp as an example:

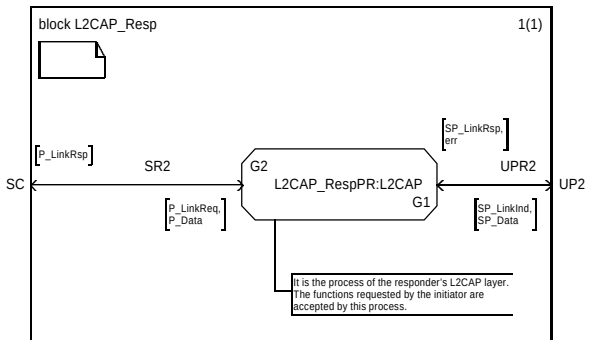


Fig 10: *L2CAP protocol block-level description particulars*

The Figure 10 sufficiently indicates that the block contains only one single process. One can easily observe the sign channels (SR2,UPR2) used in communication, their connections to the channels illustrated on system-level, and the type and character of signs transmitted by them.

Every process is manifested as a separate finite state machine that communicates with the other and thus they build up the actual system. The full-scale description capacity of SDL provides opportunity for defining processes, illustrating its structures with the help of which the stages and flow of operation can also be illustrated. I demonstrate the inner operation of L2CAP layers on this level (receiving and sending signs, actions and status shifts) and the way of realising services in the form os these layer particulars. And now let's have a look at such details of protocol SDL-description (see Figure 11)

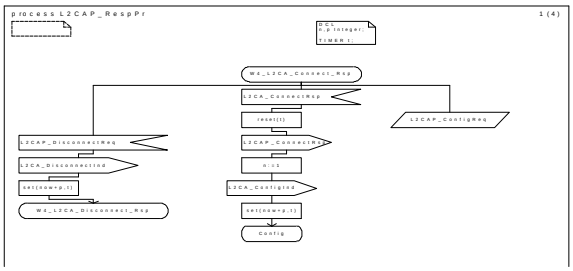


Fig 11: *Particulars of L2CAP protocol process-level description*

The highlighted part illustrates the possibilities of the slave originated from W4_L2CA_Connect_Rsp starting status.
As long as it gets L2CAP_DisconnectReq disconnection requesting PDU from the master, it informs the higher layer with the help of L2CA_DisconnectInd service primitive about the situation and after setting the timer it shifts to W4_L2CA_Disconnect_Rsp status preceding disconnection.
The description is similar in the case of receiving accepted connection response from the higher layer. The total L2CAP_RespPr process description continues on several other pages. It is necessary to similarly provide similarly the description of L2CAP_Ini block and its process and thus we get the total formal description of protocol. In the case of difficult systems we make differences between hierarchy-levels for the sake of better understanding of dividing block-levels into new block-levels. Paying attention to functionality in the field of illustration we have the opportunity of utilising user-friendly illustration by means of carefully thought arrangement with the help of SDL even in the case of huge systems.

4 Conclusion

We could see in the foregoing that the planning of Bluetooth network usage models offers the use of design patterns and the application of reuse theories. Finding models in protocol development may even take place in the phase of requirement-analysis due to the structured and documented protocol standards. In recent years protocol standardisers have more often used the well-known formal languages as unambiguous transmitting languages, namely SDL, MSC, and ASN.1. It is rather the verification of implementation that is the semantics of languages and their adequate application in planning methods.

Recent development of SDL approves that the engineering of any real-time system can be realised in SDL. The main power ever of protocol engineering derives from its basis: the role of protocol specification mixed with prevailing OO techniques ensures unique and economical planning of protocol life-cycle elements. However, the role of implementation and verification has been increasing due to the fact that by using formal languages the analysis is not a usual OMT technology analysis any longer.
One way of facilitating prevention is creating SDL packages with the application of pattern-based programming and applying the OO characteristics and specification of SDL. The SDL-2000 offers new possibilities with the introduction of agent type. Actions (that can be granted on system-level), definable variables and several block instances are such powers that reinforce pattern-based planning and with their help even new protocol patterns can be introduced.

References:

[1] ITU-T Recommendation Z.100 (11/99), *Specification and Description Language (SDL) Supplement 1 to Z.100 (05/97): SDL+ Methodology. Update for SDL-2000 in progress* . ITU-T, 2001.
[2] Jan Elsberger, Dieter Hogrefe and Armadeo Sarma, *SDL Formal Object-oriented Language for Communicating Systems*, Prentice Hall, 1997
[3] Zoubir Mammeri, *SDL modélisation de protocoles et systèmes réactifs*, Hermes Science, 2000
[4] http://www.sdl-forum.org/Tools/index.htm
[5] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*, John Wiley&Sons, 1996
[6] Juha Pärssinen and Markku Turunen, Patterns for Protocol System Architecture *In PloP 2000 Proceedings*, http://jerry.cs.uiuc.edu/~plop/plop2k/proceedings/Parssinen/Parssinen.pdf,http://jerry.cs.uiuc.edu/~plop, 2000
[7] Telelogic Tau™ 4.1 *User's Manual Guidelines*, Telelogic AB, 2000
[8] SIG, *Specification on the Bluetooth System V.1.1*, Vol. 1 and 2, www.bluetooth.com, 2001
[9] Tibor Dulai, Anna Harmatné Medve, Bluetooth - One of the Best WPAN Solutions for Bridging PAN and Wider Networks?, *14th Euromicro Conference on Real-Time Systems ,Workshop on Real-Time LANs in the Internet Age.*, June 19 - 21, 2002, http://www.hurray.isep.ipp.pt/rtlia2002/full_papers/16_rtlia.pdf, Vienna, 2002
[10] ITU-T Recommendation Z.120, (11/99) *Message Sequence Chart (MSC)* Geneva,1999.
[11] *ASN.1 Informational Site*, http://asn1.elibel.tm.fr/