SDL Design of a Radio Resource Control Protocol for 3G Evolution Systems with Two Different Approaches

Tae-Hyong Kim[1], Jae-Woo Kim[1], Qi-Ping Yang[1], Jae-Hyoung Lee[1], Soon-Gi Park[2], and Yeun-Seung Shin[2]

[1] School of Computer and Software Engineering, Kumoh National Institute of Technology, Gumi, Gyeongbuk 730-701 Korea
{taehyong, eva0191, saintwind, zzeng09}@kumoh.ac.kr
[2] Mobile Telecommunication Research Laboratory, Electronics and Telecommunications Research Institute, Daejeon, 305-350 Korea
{yoyo, shinys}@etri.re.kr

Abstract. Despite the increasing need of formal methods, people in the industry still hesitate to use them for product development because they are not sure of success with that novel approach in their own situation. In order to encourage those people we show our experience of designing a radio resource control protocol for ETRI's 3G evolution systems in SDL with two different approaches: pure-SDL and hybrid-SDL approaches. From our design and verification results, we make an empirical evaluation of those two approaches in several aspects and suggest a simple guideline for selecting an appropriate approach according to the situation.

1 Introduction

Since several formal description techniques were developed and standardized to help developing a reliable network system, a lot of work has been done to design, implement, and verify communication protocols with formal methods[1,2,3]. Among those languages the Specification and Description Language (SDL)[4] showed a remarkable success owing to the continual refinement of its syntax and powerful development tools such as Telelogic Tau[5]. Those tools provide integrated environments for the design and implementation of a distributed system with automated verification features such as trigger-based simulation and reachability-analysis-based validation. The reliability of a product is a major goal of the industry however it normally costs very much. Therefore the existence of powerful SDL tools encouraged the industry to use formal methods for the development of network products.

However, a lot of system development engineers still stick to traditional development methods with general programming languages such as C language because most of them are afraid that formal development methods will greatly increase their works in developing a real-world large and complex network system. In addition the development of current large network systems generally involves many working groups of cooperating engineers, each of which is in charge of developing a part of the whole system. Therefore, the decision to change the development method usually requires the agreement of all those engineers. That can be a practical barrier for a formal method to become popular in the industry.

Cellular communication systems evolved from the present third generation (3G) system, usually called the third generation evolution (3GE) or the long-term evolution (LTE) cellular systems, may be a good example of such a real-world large and complex system. Such a new communication system, however, is based on the previous system so usually uses quite a few features of the previous system. This means there are many of the existing components or libraries can be used in a new system with a little modification. Naturally they want to use them to reduce the cost and the risk of developing a whole system newly. Hence it may be very difficult to design such a system with a new language and a new tool.

We designed a radio resource control protocol between the User Equipment (UE) and the Universal Mobile Telecommunications System (UMTS) Terrestrial Radio Access Network (UTRAN) for a 3GE cellular system of Electronics and Telecommunications Research Institute (ETRI) with an SDL and C combined approach, which we call *hybrid-SDL* approach. This work was done as a collaborative project with ETRI in order to construct a prototype UTRAN for a 3GE system with other protocol implementations. The protocol was named Radio Resource Control Plus (RRCP) and is based on the specification of the third Generation Partnership Project (3GPP) release 6[6]. RRCP is the core protocol of UTRAN and is the most complex so we used SDL for producing a functionally correct implementation. Actually the hybrid-SDL approach was a reasonable solution because all the other protocols of that prototype UTRAN were decided to be implemented in C and the data structure and libraries constructed in C must be shared with all the protocols.

In this paper we describe design issues in modeling RRCP with the hybrid-SDL approach. In order to evaluate that design we modeled the same protocol again mostly in the SDL world. This approach we call *pure-SDL* approach. We evaluate those approaches by comparing the two designs and their verification results. After identifying the strength and weakness of those approaches, we present a simple guideline for selecting an appropriate design approach to develop real-world network systems for various situations.

Section 2 summarizes several related works that designed UMTS protocols in SDL. The target system, ETRI's 3GE-2005 system and the motivation of this work are briefly explained in section 3. In section 4 we explain two design approaches, pure-SDL and hybrid-SDL approaches, in detail. Section 5 explains overall design issues and some design details of our SDL system. Then the process and results of verification and target porting are presented in section 6. Section 7 evaluates those design approaches and suggests a simple guideline to decide an appropriate one in a certain situation. Finally we conclude this paper in section 8.

2 Related Works: Designing UMTS Protocols in SDL

The UMTS is the 3G cellular communication system that is currently serviced mostly in European countries. The 3GPP generally updates standard documents several times every year. So the UMTS protocols are good targets for the implementation with formal languages for automatic verification. There have been a couple of case studies that designed UMTS protocols in SDL, some of which we briefly introduce as follows.

P.J. Song *et al.* designed Wide-band Code Division Multiple Access (WCDMA) radio interface protocols based on the 3GPP Release 99 specification in SDL[7]. The design goal was to verify UE protocols with Tau Simulator in the SDL environment. Hence the design of WCDMA protocols focused on the UE side of the system. Modeling each block in SDL did not follow the standard structure of its corresponding protocol specified in the 3GPP requirements. The UE side protocols were designed fully in SDL including the Abstract Syntax Notation One (ASN.1) encoding and decoding. But that SDL design had not been ported to a specific target system. It could be free from additional C coding because it is a stand-alone system in the SDL environment only and the performance was not the main design issue. The functional behavior of the SDL design was verified successfully by simulating that design with Tau Simulator. Afterwards another team of ETRI designed a beyond-3G (B3G) system based on the 3GPP Release 5 specification[8]. They implemented a radio control protocol for the access network side, which is called 'RC', and it corresponds to Radio Resource Control (RRC) in the 3GPP specification. They used SDL for modeling the system but most of the functional behaviors were implemented in inline C-code and external C libraries because one of the main goals was a porting of the system to a real platform, the VxWorks system[9]. Accordingly not much effort had been done in the SDL design and its verification.

R.J. Skehill *et al.* at the University of Limerick developed distributed UMTS signalling layers in SDL to construct a testbed for the Information Technologies Programme (IST) Advanced Radio Resource management fOr Wireless Service (ARROWS) project[10]. They used the so-called SDL Object Modeling Techniques (SOMT) for the effective system design, which is now a general formal design technique using SDL and its tool. Two UMTS signalling protocols, RRC and Radio Access Network Application Part (RANAP), and Non-Access Stratum (NAS) drivers for UE and the Core Network (CN) were designed together in an SDL system for the verification of UMTS upper signalling layers. After the simulation and reachability analysis for the whole system with Tau, each protocol was integrated independently into Linux. They focused on signalling functions when modeling each protocol in SDL but technical solutions for fast implementation or good performance seem to be not considered seriously.

J. Colás *et al.* at the University of Málaga designed UMTS protocol layers for the radio access interface in SDL with object oriented design techniques supported by SDL-2000[11]. They wanted to follow the structure division suggested by the specifications unless they found more appropriate solutions. They especially tried to make the design have a good reusability because communication protocols maintain many similarities along the evolution path of the systems. They were interested in the quality of SDL design and used various object oriented syntax of SDL-2000. As a consequence, their design seems to be structured well: very refined and optimized. Such a sophisticated design, meanwhile, may have low readability due to its unfamiliar syntax especially to novices of the language. In [12] they ported the layer 2 protocols to Windows 2000 and Linux to measure their efficiency with regard to the data transmission rate.

3 ETRI's 3GE-2005 System

In order to develop a reliable and ultimate 3GE system with an incremental approach, each year ETRI makes an interim design for the final system, implements, and verifies with in-house testing. The 3GE system developed in 2005 is called 3GE-2005 system and it is based on the specification of 3GPP release 6 as indicated before. This system is composed of three subsystems: the access system subsystem (ASS), the UE subsystem (UES), and the CN subsystem (CNS). Figure 1 shows the structure of ETRI's 3GE-2005 system focusing on the ASS.

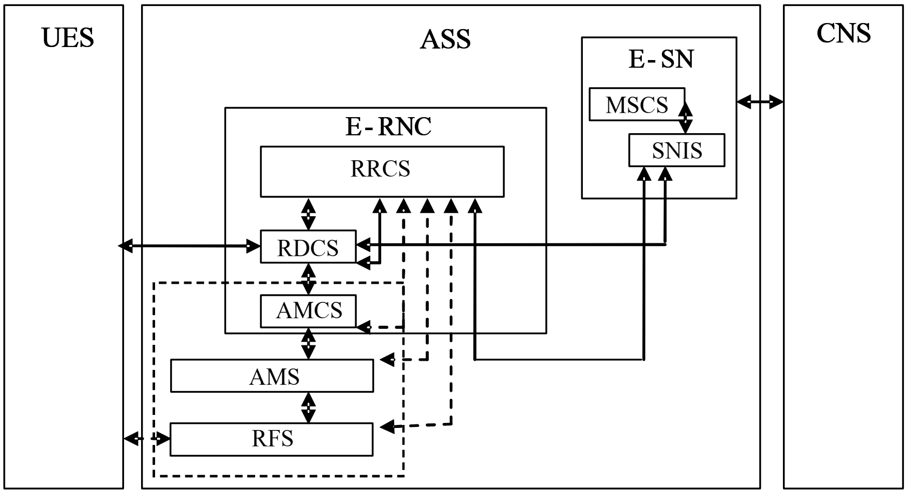

Fig. 1. The structure of ETRI's 3GE-2005 system emphasizing the ASS

The ASS contains the Evolved Radio Network Controller (E-RNC), the Evolved Serving Node (E-SN), the Access system Modem baseband Subsystem (AMS), and the Radio Frequency Subsystem (RFS). The target protocol of this paper, RRCP is for both the RRC subsystem (RRCS) in E-RNC and the peer part in the UES. The RRCS in E-RNC provides the functions and services of the existing RRC and RANAP including MBMS services, such as signalling for call processing, the radio resources management in the ASS, the radio resource control of a UE, the coordination between the E-RNC and the E-SN, and the message routing between the E-SN NAS and the UE NAS.

The 3GE-2005 system was designed with a general top-down approach. The team designed the structure of ASS by analyzing the requirements of that system, decided the order of message exchanges between modules in the subsystems with Message Sequence Charts (MSC) for each functional process, designed the data structures for messages, local databases, and interfaces, and finally implemented functions of each module. When implementing and verifying modules of the system they used a traditional implementation method with C language except one module, RRCS. Because RRCS is the most complex part in the system, they wanted to use SDL in designing RRCS for obtaining a functionally correct implementation with formal verification techniques supported by SDL tools. In order to make the most use of powerful verification techniques of an SDL tool, however, not only RRCS but other modules in ASS and in UES must be included in the SDL world. For this reason they designed an SDL system for the whole system including both ASS and UES. Design issues of those additional modules are discussed in section 5.1. Note that the main goal of that SDL system is to make a functionally correct RRCS implementation. Figure 2 shows the top-level structure of that SDL system which contains a UTRAN block and three UE blocks for functional testing of RRCS at the multiple-UE's condition. It also contains additional three blocks: two for testers and one for emulating broadcast transmission by air.

4 Two Design Approaches: Pure-SDL and Hybrid-SDL

The pure-SDL approach tries to only use SDL and other languages directly supported by an SDL tool in designing a system for the maximal formality. Figure 3 shows the flow of constructing a program by Telelogic Tau with the pure-SDL approach. It tries to use no external C libraries and no external C-code including header files except environment functions.

On the contrary the hybrid-SDL approach freely uses external C-code and header files according to the given situation. The flow of constructing a program by Tau with the hybrid-SDL approach used in our work is shown in Figure 4. Note that we used ASN1C compiler[13] instead of ASN.1 utilities of Tau because the data structure in C produced by ASN1C had to be shared with developers in charge of other parts of ASS. The source code produced by ASN1C consist of type definitions and encode/decode functions. ASN1C also provides a run-time library required to use those encode/decode functions.

As described in the introduction, the hybrid-SDL approach is a reasonable solution in the situation such that data structures or libraries written in C are used in common. In addition, system developers that are not used to designing in SDL are likely to give a preference to this approach when he has to use SDL. In this situation this approach will probably take less time in designing than the pure-SDL approach. Furthermore they need not stick to SDL tools; they can use other tools or libraries freely for better performance. However we can expect that the power of the SDL tool in use may be limited and that there may be some potential problems in the implementation due to the integration

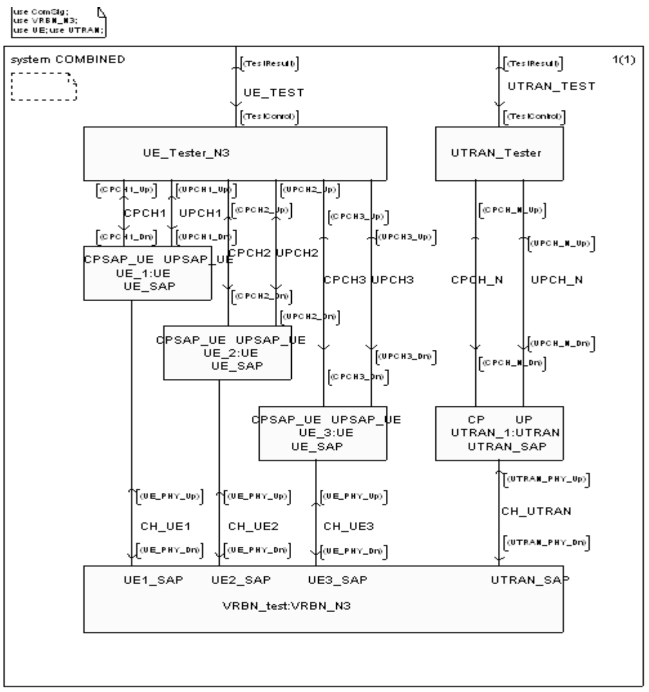

Fig. 2. The top-level structure of the SDL system designed

of heterogeneous modules. We examine the strong and weak points of those two approaches in detail from the results of design and verification in Section 7.

5 Designing the SDL System

This section describes design issues of the SDL system shown in Figure 2. Some points for the overall design are explained and then a part of design details follows with the hybrid-SDL and pure-SDL approaches.

5.1 Overall Design

Recall that the design goal of our SDL system is to build a functionally correct RRCS implementation. In order to satisfy that goal we drew up the global design as follows. First, block division and channel structure of the system were designed according to the specification suggested by the 3GPP standards for the sake of high readability and reliability. We modified the structure division only we identified where it was really necessary to make changes for better performance. Figure 5 shows the top-level design of block type UTRAN. Block RRCP is for the target protocol that corresponds to RRCS, and block HMAC stands for higher Medium Access Control (MAC) and includes the protocols, Radio Link

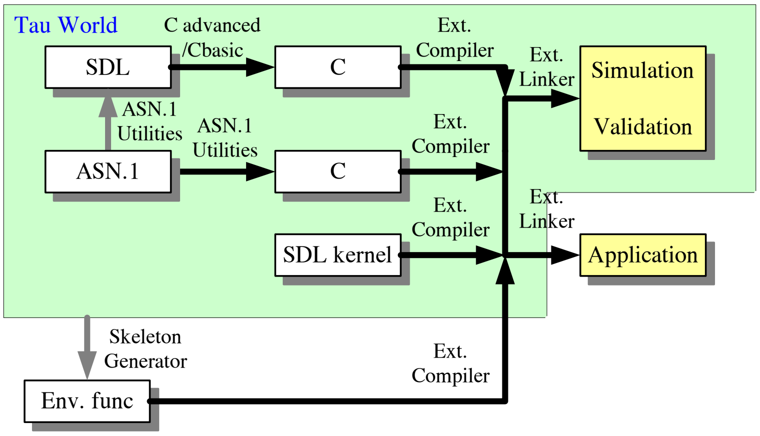

Fig. 3. The flow of constructing a program by Tau with the pure-SDL approach

Control (RLC), Packet Data Convergence Protocol (PDCP), and Radio Packet Tunneling Block (RPTB). The channel structure between RRCP and HMAC, and between HMAC and MAC also follows the specification of service access points (SAP) and logical channels of the standards.

The SDL design of the target protocol, RRCP in UTRAN is shown in Figure 6. According to the standards, RRCP includes four entities for its functional behavior, Dedicated Control Function Entity (DCFE), Broadcast Control Function Entity (BCFE), Paging and notification Control Function Entity (PNFE), and Shared Control Function Entity (SCFE). Transfer Mode Entity (TME) and Routing Function Entity (RFE) handles the mapping and routing of messages between different entities respectively. We added Control Routing Function Entity (CRFE) for the routing of lower layer configuration messages between some function entities in RRCP, DCFE, BCFE, and PNFE, and lower layer protocols because those functional entities exchange several pairs of those configuration messages with each of low layers.

The second point of the overall design is for the lower layers, HMAC, MAC, and PHY. Actually the porting of those protocols to a real platform is the work of other team who did not use SDL so we did not have to implement complete functions of those layers. For protocols inside HMAC: RLC, PDCP, and RTB, we designed their structure according to the standard specifications. Those protocols manage messages with a separate process for each connection according to the transmission mode, e.g. acknowledged mode (AM) or unacknowledged mode (UM). However we left out detailed massage manipulation functions such as segmentation and reassembly, or ciphering for their future completion for direct targeting from SDL designs. For the remaining lower layers, MAC and PHY, we implemented the minimum functions required to pass messages correctly and to exchange controlling messages with RRCP according to the specifications. The modeling effort of those additional blocks took much less time, compared with that of RRCP, because they don't have much detailed processing to do.

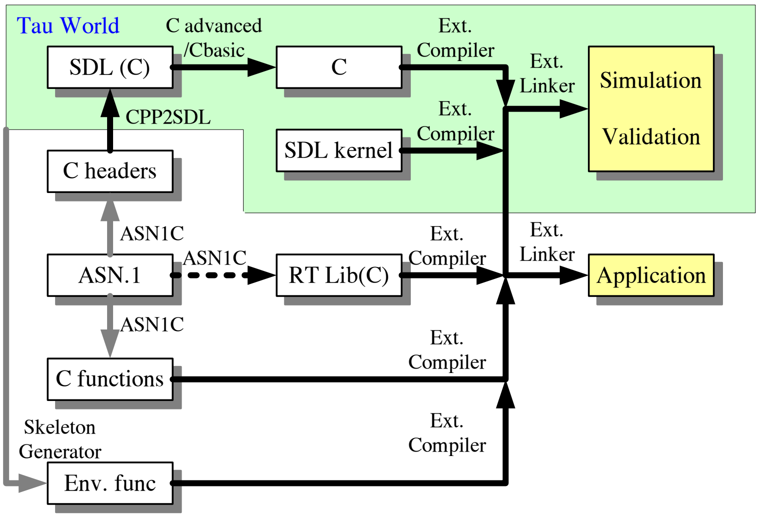

Fig. 4. The flow of constructing a program by Tau with the hybrid-SDL approach

The SDL system shown in Figure 2 contains two tester blocks for UE's and UTRAN respectively. The main goal of these tester blocks is to send appropriate triggering or response messages for testing UE's and UTRAN. Block UTRAN_Tester also includes a NAS simulator for UTRAN because NAS exists not in UTRAN but in CN. It exchanges RANAP and MBMS session management messages with RRCP according to the specification. Block VRBN indicating virtual radio broadcast network is used to broadcast messages from UTRAN to all UE entities because Tau does not support broadcast transmission with the phrase '`VIA ALL`'.

Finally, we tried to use simple syntax instead of complex object-oriented features supported by SDL-2000 for simplicity and reliability except some block types for increasing reusability. Note that a major advantage of SDL is that it is easy to learn, read and write, especially for beginners, owing to its intuitive diagram and simple grammar. At first we wanted to use some object-oriented syntax for reusability and systematization such as state aggregation but we had to tiresomely check if Tau supports those syntax. The available version of Tau, 4.6.3, unfortunately, did not support many of SDL-2000 features. For the features that are required but not supported by SDL such as pointers we could use the special packages or libraries offered by Tau. Those were very useful for complicated function implementation but their incompleteness also became the problems for modeling.

5.2 Design Details for Message Handling

We skip the detailed design of each block and process on account of the limited space. Note that we tried to increase the readability and scalability of our design

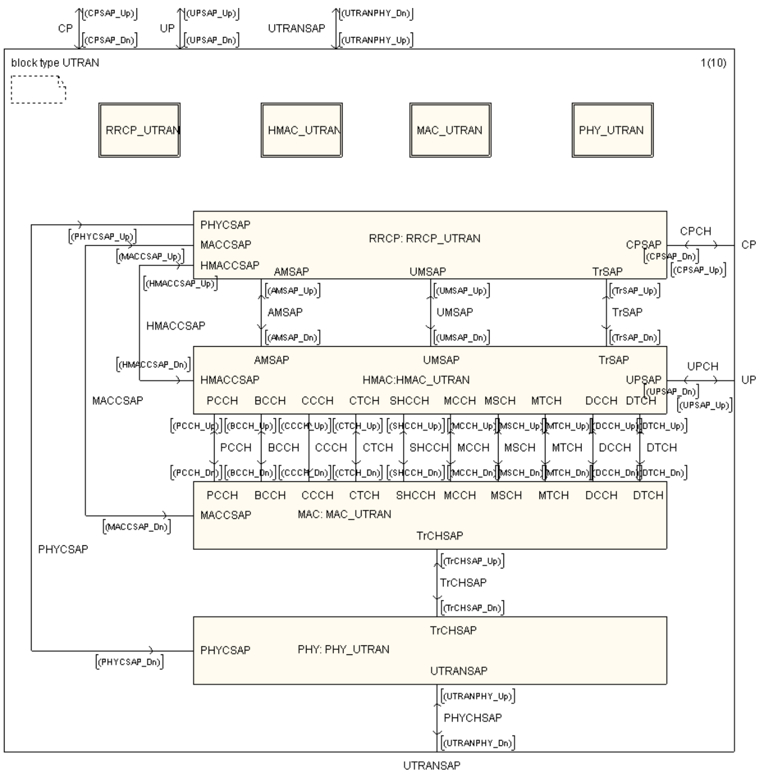

Fig. 5. The top-level design of block type UTRAN

and we regretted that we could not use composite states in our design because they are not supported by Tau. In this section we describe only a part of design for RRCS message handling with the hybrid-SDL and pure-SDL approaches.

With the hybrid-SDL approach, we used the ASN1C compiler to handle the RRCP definition in ASN.1 which was obtained from a slight modification of the standard RRC definition[6]. ASN1C generated some C header files for the data structure and some C functions for encoding and decoding of the data in class definition, protocol data unit (PDU) definition, and information elements from the RRCP definition. These functions provide application programming interfaces (API) to handle RRCP messages. Figure 7 shows how RRCP handles a message for radio resource management when it sends and receives that message respectively. When sending such a message, RRCP initializes the message contents, encodes them in ASN.1, constructs the message, and finally sends it. Receiving process of a radio resource management message is in the opposite order; RRCP decodes the ASN.1-encoded part, extracts the message type, controls radio control procedure, and finally stores the message contents in its local database.

In order to process messages efficiently, we created C libraries, 'make', 'pack', 'unpack', and 'store' for making message contents out of information elements

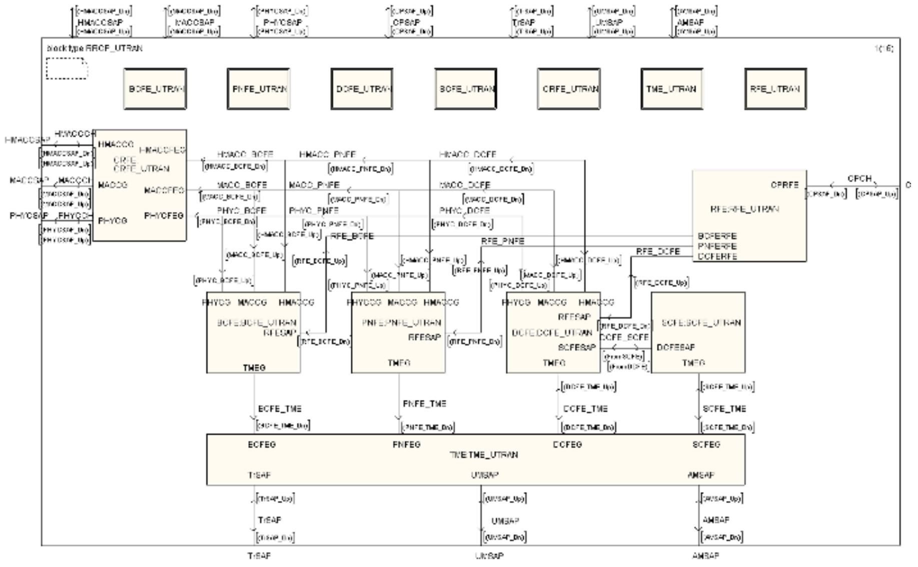

Fig. 6. The top-level design of block type RRCP

stored in the local database, for encoding them in ASN.1, for decoding the message contents encoded in ASN.1, and finally for storing information elements required for call and MBMS processing in the local databases. For example, the function `fnRrcp_make_msgInitialDirectTransfer()` fills up the contents of 'Initial Direct Transfer' message with appropriate values and the data from information elements stored in the database named 'callInfo' for the given condition. In addition, we created another C library, 'utility' for extra functions to process some miscellaneous work, e.g. `fnRrcp_initialize_mbmsInfo()` to initialize the database named 'mbmsInfo'.

With the pure-SDL approach, the built-in ASN.1 utilities of Tau are used to handle the data in ASN.1. The data structures in the RRCP specification in ASN.1 were transformed automatically in SDL by those utilities. Local databases were also transformed in SDL. Instead of C libraries we created the corresponding SDL procedure for each function included in those libraries. We also used inline C coding supported by Tau where the modeling is difficult with SDL syntax only.

6 Verification of the Design

In order to verify the design of RRCP, we used simulation, validation with reachability analysis, and testing on target. We used both SDL models designed with the hybrid-SDL and the pure-SDL approaches. Figure 8 shows the verification process we used.

First, we checked the functional consistency of the design with the test scenario in MSC by Simulator UI of Tau at both single-UE and multiple-UE's con-

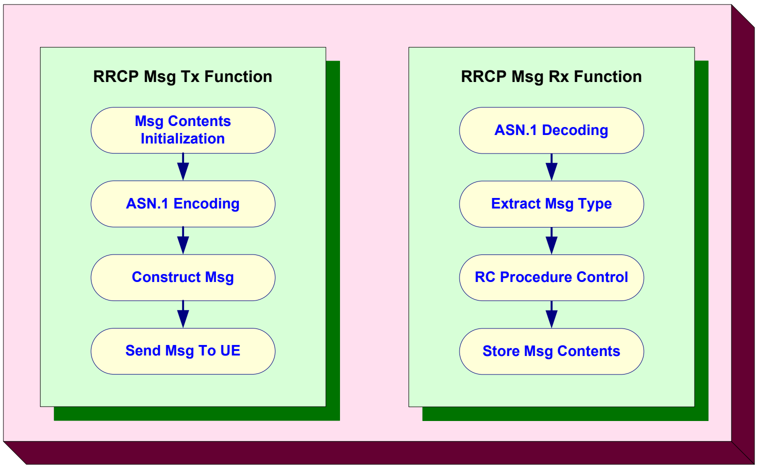

Fig. 7. Message handling procedure of RRCP for radio resource management messages

ditions. That test scenario was designed to check if the functional behavior of the SDL model matches the design requirements of RRCS in normal situations. During the simulation at the single-UE condition, there was neither mismatching of logic flows nor semantic errors in SDL design for either model. Verification of the design at multiple-UE's condition is necessary to check if the resource management functions of RRCP in UTRAN handle each of the UE's without any problem. In addition, the system may do wrong actions due to some signal racing problems caused by the messages sent by multiple UE's. Those errors are usually due to the incorrectness or incompleteness of the design that cannot be easily identified during the simulation at the single-UE condition. Fortunately there was no error found during that simulation for either SDL model.

Next, we made a couple of reachability analyses of the design with Validator UI of Tau for the multiple-UE's condition. We skipped validation for single-UE condition from the experience with simulation. Validation of the whole system requires a lot of time and memory due to its huge size and high complexity so we decided to decrease the scope of validation to the 'attach' process only. We used two reachability analysis techniques supported by Validator UI, bit-state and tree walk explorations. In case of the model designed with the hybrid-SDL approach, we unfortunately failed to obtain the result due to unexpected run-time errors. Those errors say that the program failed to read the value at a specific location of system memory. From several experiments to clear the cause, we found they are related to the libraries generated by ASN1C compiler. Thorough exploration of the global state of the system might cause a memory crash due to imperfect integration with C libraries that cannot be found in ordinary situation. With the model designed with the pure-SDL approach, we had not experienced any run-time errors. During that validation, we found some weaknesses of the design that can produce errors in very exceptional situations

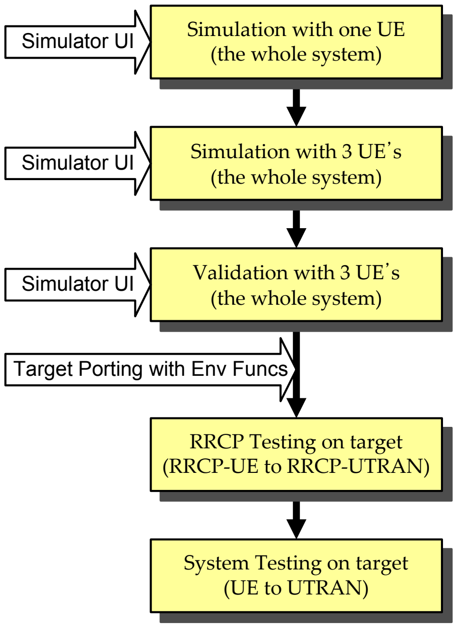
Simulator UI
Simulation with one UE (the whole system)
Simulator UI
Simulation with 3 UE's (the whole system)
Simulator UI
Validation with 3 UE's (the whole system)
Target Porting with Env Funcs
RRCP Testing on target (RRCP-UE to RRCP-UTRAN)
System Testing on target (UE to UTRAN)

Fig. 8. The verification process

such as extremely long transmission delay of a specific message which may cause the signal racing problem. Owing to that validation we could obtain a more reliable implementation by eliminating those weaknesses.

After verifying the design by simulation and validation, we ported each RRCP module in UE and in UTRAN to the Linux platform by completing the environment functions required for the target integration. The socket interface was used for the communication with other modules. Before system testing between UE and UTRAN, preliminary testing between two Linux-ported RRCP modules was performed to increase the possibility of its successful operation during the final system testing. In system testing, only the RRCP module for UTRAN was used to generate a UTRAN instance; UE instances were created from the SDL model designed by other team. The goal of this testing was to verify the inter-operability of UE entities and the UTRAN implementation developed by different teams; the efficiency of implementations was out of interest this time. Fortunately the testing on target was successful; there were no particular errors except a couple of trivial ones due to some configuration problems. We note that the testing on target was performed entirely by ETRI.

7 Comparative Evaluation of Two Approaches

From the result of design and verification, we evaluate the two design approaches in various aspects and suggest a simple guideline for selecting an appropriate one according to specific conditions.

First, we would like to show the development time of our SDL system with two approaches in Table 1. Each number in parentheses indicates the number of members involved. Our team was composed of 6 engineers; two were SDL experts, another had some experience with SDL, and the others had no experience with SDL. All members were good at C programming except one. Actually we could not start designing with two approaches at the same time due to our tight schedule. First we designed with the hybrid-SDL approach and then with the pure-SDL approach because we had some C-code of the previous system that was a basis for the target system. Therefore Table 1 is not a fair comparison; it just shows our result.

Table 1. The development time of our SDL system with two approaches

	Hybrid-SDL approach	**Pure-SDL approach**
Training	2 weeks (4)	1 week (3)
Implementing libraries	2 months (4)	1.5 month[1] (3)
Modeling SDL system[2]	3 months (4)	1 month[3] (3)
Verification	2 weeks (1)	3 weeks[4] (1)

Notes 1. We could save some time owing to the experience with the hybrid-SDL approach. 2. This includes debugging of the system. 3. We could use a lot of SDL-code written by the hybrid-SDL approach. 4. This includes validation time as well as simulation time.

After the development we discussed the strong and weak points of two approaches. We agreed that those points depend on the technical experience and expertness of an engineer. Also the development condition is a significant factor to give preference to one approach. Our members who were C experts but SDL novices said they preferred the hybrid-SDL approach because they could use C-code in complex functions of the system. To the other members, however, the pure-SDL approach was easier because they can use ASN.1 data structures directly in the SDL system without troublesome conversion. According to the time for learning design skills related to SDL and Tau, two approaches have both pros and cons. The hybrid-SDL approach required some time for learning how to connect the SDL system to external libraries, while it took some time to learn SDL syntax and Tau-specific features with the pure-SDL approach. The modeling speed of two approaches, according to our experience, depends on the engineers; the hybrid-SDL approach would be faster for engineers who are C experts but SDL novices, and the pure-SDL approach faster for others. As for the frequency of errors made in design and the time for correcting them, we agreed the pure-SDL approach is much better thanks to its integrated development environment. Table 2 summarizes differences between them noticed from our experiences.

According to the time required for verification and the application execution time, comparing with the pure-SDL approach, the hybrid-SDL approach took less time in performing ‘analysis ’and ‘make’. That was mainly because the source

Table 2. Noticeable differences between two approaches related to the development

	Hybrid-SDL approach	**Pure-SDL approach**
Learning issues	connecting SDL to C	Tau-specific functions
Modeling speed	faster to C experts/SDL novices	faster to the others[1]
Frequency of errors	more	fewer
Debugging time	longer	shorter

Note 1. the engineers who are C novices or SDL experts

Table 3. The strength and weakness of two approaches

	Hybrid-SDL approach	**Pure-SDL approach**
Strength	· can use the exiting libraries · may be easier to C experts · can easily collaborate with other team using data structures in C	· efficient/integrated design environment · higher readability and manageability · larger verification scope · higher reliability of the implementation · direct use of ASN.1 data structures
Weakness	· extra source management required · some coding may be overlapped · careful integration required · external tools may be used	· higher dependency of SDL tools · expertise on SDL/tools required · SDL-only coding can be difficult occasionally

Table 4. A simple guideline for selecting an appropriate design approach

Situation	**Recommendation**
· when a lot of existing C-code and libraries can be used · when collaborating with other team using C is required · when most members are SDL beginners and time is limited · when complicated data processing is often required[1]	Hybrid-SDL approach
· when a new system is entirely designed · when readability and reusability of code is important · when reliability of the implementation is the first priority · when there is enough time for learning SDL and its tools	Pure-SDL approach

Note 1. C coding is easier for complicated data processing such as complex pointer manipulations.

conversion from ASN.1 to SDL is performed each time in analysis with the pure-SDL approach. With respect to the simulation time the pure-SDL approach took more time because the simulator shows more detailed simulation results with the pure-SDL approach. In validation, as we described before, we could not compare two approaches due to the failures happened during validation with the hybrid-SDL approach. Fortunately no errors happened during execution of the implementations. The execution time will depend on the optimization of the code and the characteristics of libraries used in each approach. We narrowed down the scope of comparison to the execution time of the attach process in order to find out the cause of difference. In several experiments, two approaches did not show a significant difference in execution time. This result show that the performance of Tau with its built-in functionalities is good enough to be used in the development of real-world network systems. In order to evaluate the performance of the generated software exactly, however, more detailed analysis and experiments are required.

According to our discussion, we draw the strength and weakness of the two approaches as shown in Table 3. Table 4 is a simple guideline to developers in the industry for the selection of an appropriate design approach. We note that it was generated from our experiences and accepting this guideline as a general one requires more experiments and experience. In addition, actually our division of the design approach is rather idealistic because the pure-SDL approach is very strict and difficult to follow completely. With a right understanding of the two approaches, however, you can find a good compromise between them appropriate for your situation.

8 Conclusions

Lately the industry has taken a great interest in the reliability of products to win in the fiercely competitive market and formal approaches to the development of a product are now coming into their sight. Especially in the telecommunication area, more and more successful stories have been reported enough to encourage the industry. However people in the industry still seem to hesitate to apply formal methods in development because they usually don't know well how to start with a new approach and how to migrate from the existing approach. Sometimes they want to use a formal approach partly in a specific condition as a trial. But they usually don't have enough information that will be direct help to their development.

This paper showed an experience in the development of a network protocol, RRCP for ETRI's 3GE systems, with two different design approaches: the hybrid-SDL and the pure-SDL approaches. We also draw the strength and weakness of those two approaches in several aspects from our experience and present a simple guideline for the selection. Actually the hybrid-SDL approach can be a practical solution when you migrate from the traditional approach with C language or when you have to share the data structure or C libraries with other team. But you should be careful in using external tools or writing external C-code to obtain a reliable implementation because imperfect integration of SDL and C code may cause unexpected run-time errors. The pure-SDL approach gives higher readability, reliability, manageability, and verification capability than the hybrid-SDL approach in general and you can easily handle all sources in the integrated environment provided by the SDL tool. Hence the pure-SDL approach is recommended when you start to design a new system entirely. You can also find a good compromise between the two approaches which is appropriate for your situation. To derive detailed criteria to decide design methods for various situations, further systematical analysis of design approaches should be performed.

The International Telecommunication Union (ITU) Telecommunication standardization sector (ITU-T) now has an objective to integrate its standard languages such as SDL, MSC, and ASN.1 using Unified Modeling Language (UML) 2[14] as a framework and defining UML 2 profiles for those languages[15]. Owing to the enhanced features of version 2 such as formal syntax added and powerful commercial tools supporting UML 2, UML seems ready to be an excellent tool in the formal development of a general system. In order to encourage the industry to come into this formal world, we hope for a lot of practical experiences in various situations with UML also.

Acknowledgements

This paper was supported in 2005 by Research Fund, Kumoh National Institute of Technology and Electronics and Telecommunications Research Institute. We also thank Dr. Daniel Amyot and the anonymous reviewers for their valuable comments.

References

1. P.R. James, M. Endler, and M.-C. Gaudel, Development of an atomic-broadcast protocol using LOTOS. Software - Practice & Experience, Vol. 29, Issue 8, pp.699–719, John Wiley & Sons, Inc. 1999.
2. P. Amer, A. Sethi, M. Fecko, and M. Uyar, Formal design and testing of army communication protocols based on Estelle. Proc. of the 1st ARL/ATIRP Conference, College Park, pp.107–114, 1997.
3. B. Hatim, M. O. Droma, Telecommunication software development using SDL-92: practical experience. The 2nd IEEE Int'l Conf. on Engineering of Complex Computer Systems (ICECCS'96), pp.273–277, 1996.
4. ITU, Recommendation Z.100, Specification and Description Language (SDL). ITU, Geneva, 1999.
5. Telelogic AB Inc., Telelogic TAU Generation 1 SDL Suite Ver.4.6, 2005. See http://www.telelogic.com.
6. 3GPP, Radio Resource Control (RRC) protocol specification. 3GPP TS 25.331 V.6.5.0, 2005.
7. P.J. Song, M.H. Noh, and D.H. Kim, Design and Implementation of W-CDMA Radio Interface Protocols Using SDL Development Environment. CIC 2002, LNCS 2524, pp.442-452, Sringer, 2003.
8. ETRI, HMm, Technical Specification Radio Access Research Team, 'Radio Control (RC)'. HMm SPC-0310-250.200, 2004.
9. Wind River Systems Inc., Wind River Platform for Network Equipment, VxWorks Edition. See http://www.windriver.com.
10. R.J., Skehill, I. Rics, and S. McGrath, SDL System Development of Distributed UMTS Signalling Layers. 2nd Annual ICT Information Technology and Telecommunications, 2002.
11. J. Colás, J.M.Perez, J.Poncela, and J.T. Entrambasaguas, Implementation of UMTS Protocol Layers for the Radio Access Interface. SAM 2002, LNCS 2599, pp.74-89, Sringer, 2003.

12. V. Morillo-Velarde, J. Colás, J. Poncela, B. Soret, and J.T. Entrambasaguas, UMTS Protocol Development using Formal Languages. Proc. of the IASTED Int'l Conf. on Communication Systems and Networks, pp.274–279, 2004. Entrambasaguas Proceeding IASTED Communication Systems and Networks, Spain, 2004
13. Objective Systems Inc., ASN1C - ASN.1 to C/C++ Compiler, Ver.5.3, 2002. See http://www.obj-sys.com.
14. Object Management Group, The Unified Modelling Language Version 2.0, 2004. See http://www.uml.org.
15. Rick Reed, The chairman's report. The Annual Meeting of the SDL Forum Society, 2005.