VERIFICATION OF SOFTWARE RECONFIGURATION PLATFORMS

Stoytcho Gultchev, Klaus Moessner, Rahim Tafazolli
Mobile Communications Research
Centre for Communication Systems Research,
University of Surrey Guildford, Surrey, UK - GU2 7XH
(S.Gultchev, K.Moessner, R.Tafazolli@surrey.ac.uk)

ABSTRACT

SDR technology has significantly matured and it becomes an increasingly important tool enabling the interlinking between, and the interoperation of, different wireless technologies. SDR technologies facilitate easy to use and they support adaptive communication platforms. However, the standard compliance and system's correct functioning are generally the most critical problems and they need to be ensured. Addressing compliance and correct functionality, software and configuration verification and validation techniques are required. These techniques will need to be agreed and standardised to facilitate interoperability between different types of reconfigurable platforms and systems. This paper provides an overview of the influencing factors and shows some direction and examples of how such interoperability can be achieved.

1. INTRODUCTION

Business models for commercial implementations of SDR based radios are emerging and companies aim to specify and implement the immediate requirements into coherent system architectures. These 'reconfigurability architectures' are defined to handle download, control and installation of the targeted radio configurations and have the main aim to support mechanisms, which ensure the correctness of the targeted configurations (i.e. a configuration being the HW-SW bundle implementing a radio). Any such software installation will be prone to the usual issues like glitches, viruses, security threats etc. Secondly, the verification of these configurations and their (potentially destructive) effects on the radio emissions and, if something goes wrong, the question of the responsibilities has been widely discussed. Thirdly, the network operators will be concerned about how to ensure that a SDR node is properly installed and won't negatively affect or bring down their infrastructure, or create interference to neighbouring or co-located systems.

There are different possibilities to approach this verification of configurations; in the short term, the different manufactures may continue following the R&TTE directive and will verify and self-certify the different configurations their HW platform can implement. However, in the long term, it is conceivable that third party SW providers and uncountable combinations of software modules will exist and configurations will take place outside the control of the original manufacturer. To be able to facilitate standards conformance and to provide a testing regime, which ensures that the system specifications of the target RAT and standard compliance are met, have to be implemented.

This can be captured and achieve in an unified system design approach which considers, already during the design phase, the issue of verification and validation of the single software modules, but also of the complete configurations. This uses a design suite that employs UML for the overall system and functionality definition, and SDL for the design of the information flow and verification mechanisms of the system. The paper presents a description of this approach as well as the means and mechanisms defined for the verification of radio configurations. The paper highlights and discusses the aspects of design, implementation and verification of reconfigurable radios and will complement the reconfiguration plane concept that was introduced with the RMA (Reconfiguration Management Architecture) framework.

2. FACTORS FOR RECONFIGURATION VERIFICATION

Manufacturers will ensure the correctness of the initial SDR architectures and reconfigurable systems at the time of delivery. In the long run, due to the expected fragmentation of the market (new manufacturers, new software vendors) and the increase in availability of reconfigurable systems, there will be a need to deploy new procedures for verification of mixed configuration. This complexity of configurations cannot be just classified in a specification as research in E2R class marking approach [1]. It will require much more to reflect the whole reconfiguration validation process which may be based on the class mark mechanism but even configurations outside such framework may be implemented in reconfigurable nodes; then testing the configurations, heir implementations in the network, with

the necessary platform interaction for control and management will be part of the verification process. Variety of factors will influence this validation process but the most important will be portability, interoperability, spectrum efficiency and compatibility between platforms and reconfigurable systems. The result of a complex interaction between different players as part of it will only be achieved by standard compliance of a common reconfigurable framework. Addressing this process, there are needs for:

- Common interface definition for a reconfigurable framework [2];
- Standard compliance to the functionality of reconfiguration plane [3] - the implementation of it is not just of the reconfigurable nodes (terminals, BS, AP etc.) but also the support of the network reconfiguration service provision are the key factors;
- New tools for system software reconfiguration modules' development for compliance not just to a particular platform (manufacture) but inter-platform compatibility;
- Common reconfiguration procedures of process handling download, control and manage reconfigurability. The know-how of this reconfiguration plane implementation will still stay with different manufactures on the bases of better or more efficient way when using memory or processing resources;
- Deriving standard comprehensive verification mechanisms for supporting reconfiguration nodes i.e. generic support mechanisms for the reconfiguration service provisioning operators to have conformance, availability and radio compliance system validation procedures. This will involve:
  - A **standards (RAT) validation procedure** in the network side of the reconfiguration node's modem and system components; Again, the requirement is the procedure and interfaces to be common for everyone (standardised) and the implementation can be specific to the manufactures of equipment, giving them freedom to compete;
  - A **reconfiguration service provisioning** to be complied with specific (generic) network topology architecture [4] that will enable the open provisioning of services even from 3rd party providers. Standardisation of such architecture will be the optimum requirement;
  - Ensuring **the reconfiguration node security** and security provisioning support [5]. This will enable the possibility of reconfiguration of security features like random generators, hash functions, ciphers etc. without changing the execution algorithm of the security framework. This reconfiguration procedure requires some further investigation and also possibility of standardisation – interfaces, processes and means of communication exchange protocols. Some work in this direction has been done in the E2R project [6]. Such an important reconfiguration procedure will require verification and validation process techniques to ensure the intact of the reconfiguration security system of the reconfigurable nodes.

All these factors lead to a need for specification of the verification and the validation procedures in all the above cases.

The achievement of such a goal where software reconfigurable nodes fully comply when reconfigured will need a systematic approach for every area mentioned above with consensus between reconfiguration service and equipment providers on a generic overall reconfigurable architecture.

3. METHODS FOR VERIFICATION OF RECONFIGURABLE ARCHITECTURE AND SYSTEMS

Formal description techniques (FDTs) have a most prominent role in the development life cycle of distributed system, especially telecommunication systems. FDTs were developed to ensure unambiguous, concise, complete and consistent specification of the system under development. FDTs allow for partial or total automation of many analysis and synthesis activities in the development life cycle.

From the formal specification of user requirements to implementation, activities such as the validation of the design specification against requirement specification, the verification of design specification, stepwise refinement of formal specification towards implementation, test case generation, etc., have to be partially automated. A formal protocol definition or specification facilitates checking a protocol for logical self-consistency (validation) or demonstrating that the protocol has various desirable properties (verification). The SDL (Specification and Description Language) is one of the choices as a formalised language that is able to create clear models, which can be automatically checked for accuracy and completeness. It also offers the opportunity to use the verification and the validation in early software development phases, rather than debugging in an implementation stage. One of the real benefits of using the SDL is that a well-specified model can be simulated. The simulation is usually displayed as a dynamic MSC (Message Sequence Chart).

Object-oriented design is a design strategy where system designers think in terms of objects instead of operations or functions. The system is made up of interacting objects that maintain their own local states and provide operations on that state information. An object is an entity that has a state and a defined set of operations, which operate on that state. The state is represented as a set of object attributes. The operations associated with the object provide services to other objects (clients), which request these services when some computation is required. Object-oriented analysis

(OOA) is a well-known and popular technique for understanding a problem and analysing a system. Among many different versions of the OOA methods, the UML (Unified Modelling Language) from the OMG (Object Management Group) has been extensively accepted as a standard language for object-oriented methods and tools.

Simplicity is one of the major benefits of OOA notations. A set of class diagrams can describe complex relationships between objects from different points of view in a simple graphical way. Another major benefit is that the concepts, such as aggregation, inheritance and association, have a fairly abstract definition, which makes it possible to describe the problems in a high and abstract level in analysis situation. The general process for object-oriented design has a number of stages:

- Understand and define the context and modes of use of the system;
- Design the system architecture;
- Identify the principal objects in the system;
- Develop design models;
- Specify object interfaces.

In fact, all of the above activities can be thought of as interleaved activities that influence each other through the design process.

When documenting a model design, a sequence diagram is needed for each significant interaction. A state machine model should be provided to show how the object instance changes state depending on the messages that it receives. It is not usually necessary to produce a state chart for all of the objects. Many of the objects in a system are relatively simple objects and a state machine model would not help implementers to understand these objects.

An important part of any design process is the specification of the interfaces between the different components. Designers should avoid representation information in their interface design. Rather the representation should be hidden and object operations provided to access and update the data. If the representation is hidden, it can be changed without affecting the objects that use these attributes. This leads to a design, which is inherently simple to maintain [7].

The SDL is an object-oriented formal language defined by the ITU-T for specification of complex, event-driven, real-time and interactive applications involving many concurrent activities that communicate using discrete signals. The SDL's general adoption is partly because of its intuitive graphical notation and excellent tool support. One of the main perceived benefits of SDL over other notations such as the UML is the ability to model and reason about, e.g. via model checking tools, detailed behavioural specifications, including real-time behaviours [8].

The strength of the SDL is its ability to describe the structure, behaviour, and data of a system. The most important characteristic of the SDL is its formality. The semantics behind each symbol and concept are precisely defined. SDL has successful track record in terms of support for design, formal verification and code generation, especially for distributed, reactive and real-time applications.

The UML does not specify system behaviour in the same detail as the SDL. Process diagrams are not part of the UML, for example. However, the SDL and the UML can be complementary to each other. The UML is frequently used at the software architecture and design stage, while the SDL is now more frequently used in the more detailed process design stage. The SDL tools which provide code generation are also used in the final coding stage of software development.

The MSCs are suitable descriptions of the functional exchanges but they act also as a basis for test case development. The SDL is used at the design stage for the description of the functional behaviour and the architecture of the target system. Since the MSCs and the SDL specification are often developed independently from each other, the SDL specification has to be validated against the set of MSCs given at the requirement stage, to ensure consistency between requirement stage and design stage. The MSCs are used for requirement definition, as an overview specification of process communication, as an interface specification, as a basis for automatic generation of a skeleton SDL specification, for simulation and consistency check of SDL specification, as a basis for selection and specification of test cases, for documentation, for object oriented design and analysis (object interaction). Within the system development process, the MSCs play an important role in nearly all stages complementing the SDL in many respects. On one side, the MSC and the SDL diagrams describe the same behaviour from two different perspectives. The SDL shows how each communicating entity behaves, while the MSC diagrams show how they interact by exchanging messages. Since the MSC diagrams are easier to read, they could be helpful to both developers of the SDL specifications and their readers. On the other side, the SDL processes and the MSCs can be looked at as two different kinds of system representation which are complementary in many respects. The SDL provides a clear and comprehensive behaviour description within the individual SDL processes, whereas the communication between several processes is represented in a fairly indirect manner and thus the description of the communication behaviour in the SDL for many purposes is not sufficiently transparent. Contrary to that, the MSCs focus on the communication behaviour of system components and their environment by means of message exchange.

4. EXAMPLES OF VERIFICATION TECHNIQUES

The benefit for a development project is that the analysts and developers all can use the notations best suited for each phase of a development project.

- Use cases analysis. The purpose of the requirement analysis is to analyse the problem domain and the requirements on the system to be built, essentially by analysing the system as a black box in its intended environment;
- System model design. The purpose of the system design phase is to precisely define the architecture of the system including the detailed interfaces between different parts. In the system design the architecture of the system is also analysed in terms of implementation strategies and decomposition into work packages for different development teams;
- Specification and Implementation. The purpose of this phase is to create the executing application that implements the requirements of the use cases.

The object oriented analysis strength when analysing requirements and creating conceptual analysis models is combined with the strong back-end given by SDL tools for design, verification and code generation. Once the scenarios are properly modelled by SDL, they can be simulated and validated for the object communication protocol of the combined implementation of different scenarios.

In this work, state space exploration and state transition diagrams are used for validation of the design specification, and this is a well-known technique for automatic analysis. The state space of the SDL system of the software terminal model is explored with powerful methods and tools that will find virtually any kind of possible run-time errors that may be difficult to find with regular simulation and debugging techniques. After the errors and design ambiguities are discovered, the SDL system specification is adjusted and corrected for clearly describe system behaviours. It models the terminal's reconfiguration scenarios, including complete reconfiguration and partial reconfiguration. The process of detection and validation of implied scenarios can be used to iteratively drive the completion of scenario-based specification for the terminal model development.

For the software terminal model is used a SDL Validator to do the validation task. The Validator operates on structures known as behaviour trees or reachability graphs. A behaviour tree is a tree structure that represents the behaviour of an SDL system. The validation intends to support engineers involved in development of specifications or designs using SDL. It provides an automated fault detection mechanism that checks the robustness of the application and finds inconsistencies and problems in an early stage of development. When verifying the system against requirements, there is performed automatic verification of the requirements expressed using the MSCs, which are developed during scenario-based requirement analysis in the design period. The set of all system states represented by the behaviour tree is called the state space of the system.

By moving around in the behaviour tree, the behaviour of the SDL system can be explored and the system states reached can be examined. By investigating the error report generated by the SDL Validator and the system state where it was generated, the cause of the error can be determined, and the SDL specification and model description can be corrected accordingly. But these kinds of errors are limited in the SDL system itself, not used by the original designers to revise the scenario descriptions and functional designs.

The reachability analysis is also applied to the software terminal model in the SDL specification. When the symbol coverage rate is less than 100%, and if the parameters of the signals are correct, there are design errors in the SDL system. These kinds of errors need the SDL system designer to debug the specification and make corrections. After the state space exploration of the behaviour tree of the SDL system specification, manually and automatically, the specification is revised and corrected according to the scenario design and use case description.

Also the state transition diagrams are used for the analysis of the system modules, checking the states and the signals for whether they show the desired results. For the purpose of simplicity and page restrictions the details of all the modules' state transition diagram from the SDL Validator, applied to the RMA reconfiguration plane [3] which show the states of the modules, and what signals trigger the modules to action and the current state changes to another state are omitted from this paper.

The SDL specification for the soft terminal model is debugged and revised according to the results from the simulation and the validation. The specification finally becomes complete and correct, with no deadlock and no starvation, at least from the SDL specification point of view. The symbol and transition coverage of the specification are both 100%, which means the state space of the system is reachable and explorable completely.

All the analysis and description of this paper are for both the complete reconfiguration and the partial reconfiguration. The complete and partial reconfiguration use cases are validated together in one system validation process. So they are not explained separately.

It has already been stated that reconfigurability will provide a variety of new features and will offer its advantages to all parties (network providers, service providers, terminal manufactures, third party software vendors and the customer), however, many problems concerning reconfiguration and reconfigurability are yet to be solved. One of the main concerns is the question how, in an open software environment, radio configurations can be tested

and validated without having to go through the tedious and time consuming type approval processes. A validation and test function has been included in the RMA design and this entity (the virtual configuration (VC) process within the AcA [9]) is defined to produce information and knowledge about the reliability of the intended radio configurations.

The VC detects possible violations of radio standards during reconfiguration sequences and validates whether the intended configuration complies to the given standards (i.e. the AcA prevents the termination of configuration 1 before establishment whether the intended configuration 2 complies to the standards).

There are different stages where the standard conformance has to be tested during reconfiguration. Starting from the upload of new configuration software modules to the software repository, through to the installation and implementation of a piece of software the validity has to be evaluated. This also includes the download of the software modules and associated rules and requires mechanisms to confirm/ensure the integrity of the downloaded code.

The GAcA takes care of the complete download of the software and (software) rules from the 3rd party provider (using a secure connection), delivers it to the AcA. These mechanisms are used for the download of rules and software from another (extra-domain) AcA. The (local) AcA checks then the completeness of the downloaded source (software rules and software) and makes sure that the software delivered to the terminal conforms to the initial configuration specification of the software provider.

Before this operation is executed the AcA performs a test of the downloaded software that confirms compliance to the initial specifications of the test case (i.e. that may be provided with the software) and compares the I/O parameters of the tested module with the margins provided (with the software). This validity information is than stored in the AcA and the terminal can download/use the module for configurations. This mechanism provides the merging point for the reconfiguration software with the standards and provides the test results together with the software and rules.

The next stage is to download the software to the terminal and to ensure that the software is correctly stored in the LSWR or CRH of the terminal (using the protocols for communication between AcA and terminal and the security protocols). This download also applies when software or rules are required in the terminal for installing and creating the tag-file respectively. A further task to ensure conformance is the protection of the terminal from fraudulent configurations, this requires that the implementation of the tag-file and rules are done within the terminal, the creation of the tag-file takes place within the terminal. This guaranties that all information required for the installation of the new configuration is available and complies to the rules imposed for standard compliance.

The third step is the testing of the tag-file, the tag file is downloaded to the VC, which in turn evaluates the tag-files compliance to the network requirements. This is one of the most important verification steps for the new terminal configuration; this has to be completed before a reconfiguration can take place within the terminal. Once the procedure is complete, the reconfiguration procedure takes place as the Configuration Manager of the CMP interprets the approved tag-file and creates the necessary RMCs for the installation of the software modules.

Finally, the installation of a module on the radio platform takes place and the terminal (CM & RMCs) performs a final test before the different radio modules are connected and the new configuration becomes active. Verification procedures may differ, depending on the type/class of reconfiguration process, hence a number of reconfiguration scenarios are applied to demonstrate the functionality of the validation procedure.

The SDL implementation formally specifies the function design of the modules in the RMA architecture, and the communication sequences between the modules. With the SDL specification, the RMA architecture is formally modelled by executable specification, which can be simulated and validated before the architecture is implemented and coded darkly. Through the SDL specification, the architecture is verified and validated in an early stage, which is valuable for complex system design and save the cost of software development.

In practice the testing and validation are closely related, and after the SDL system is debugged well enough, there is no further testing to be done. The scenarios are already tested and shown in the simulation MSCs. In the similar manner networked entities of the reconfiguration plane - RMA are evaluated for the performance of procedures and message sequences compliance. This activity also deals with the correctness and validation of the design of the mechanisms between AcAs and GAcA, which describes the design specification of the network side of RMA architecture.

5. SUMMARY

SDR technologies will be at the core of any future adaptive communication platform. On such platforms, any software installation and alteration will be prone to the usual issues of software systems (installation and compatibility problems, etc.). The verification of these configurations and their (potentially destructive) effects on the radio emissions are presented as well as the effect to the network operators. The paper presented the most important factors for reconfiguration verification and different verification techniques that need to be deployed for the uninterrupted and correct functioning of a reconfiguration node. The different verification methods have been highlighted to underline the importance of FDTs in validation of

reconfiguration systems and their reconfiguration management and control parts.

Finally, an example of the implementation and verification of reconfigurable radios and complement the reconfiguration plane concept that was introduced with the RMA (Reconfiguration Management Architecture) framework has been provided.

6. ACKNOWLEDGMENTS

The work presented in this paper has formed part of Core 2 Research Programme of the Virtual Centre of Excellence in Mobile & Personal Communications, Mobile VCE, www.mobilevce.com, whose funding support, is gratefully acknowledged.

7. REFERENCES

[1] Berzosa F, "Reconfiguration Terminal Classmarks", WWRF 14, San Diego, California, USA, 07-08 July

[2] http://sbc.omg.org/

[3] Gultchev S, Moessner K, Tafazolli R, “System Reconfigurability” SDR Technical Conference 2004, Phoenix, Arizona, USA, 15-17 November 2004

[4] Gultchev S, Moessner K, Tafazolli R, “Reconfiguration Mechanisms and Processes in RMA controlled Soft-Radios Signalling”, SDR Technical Conference 2003, Orlando, Florida, USA, 17-19 November 2003

[5] Gultchev S, Moessner K, Tafazolli R, “Provisioning of Reconfiguration Services between Different Access Networks”, published in Frequenz 58 (2004) 5-6, ISSN 0016-1136, pp.126-131, May-June 2004

[6] http://e2r.motlabs.com/

[7] Sommerville I, "Software Engineering", 6th edition, ISBN 020139815X, Addison Wesley, 2001

[8] Sinnott R, “Real-Time Systems Development with SDL and Next Generation Validation Tools”, IEEE Real-Time Embedded System Workshop, December 2001

[9] Gultchev S, Moessner K, Tafazolli R “Network Based Reconfiguration Support Services for Software Radio Terminals”, IEE 3G2003, London, UK, 25-27 June 2003