Prototyping and Formal Requirement Validation of GPRS: A Mobile Data Packet Radio Service for GSM

Laurent Andriantsiferana, Brahim Ghribi, Luigi Logrippo
Telecommunications Software Engineering Research Group,
School of Information Technology and Engineering, University of Ottawa
{andrian, bghribi, luigi}@csi.uottawa.ca

Abstract

A methodology and an experience for validating a substantial part of a mobile data standard, ETSI's General Packet Radio Service, is presented. The standard was specified in LOTOS, which provided a formal prototype for the system. Testing processes were composed with the specification, and temporal logic properties were checked. At least two major design errors were identified.

1. Introduction

The application of formal methods is becoming a crucial step in detecting design flaws and in validating the requirements of complex systems. Formal methods can be used in the specification, development and verification of systems to increase the confidence in their quality and reliability.

Formal methods have witnessed a growing interest in the past decade in the areas of prototyping, simulation, and validation of informal requirements. They are increasingly used as means to build confidence in designs. LOTOS [1][2] is a formal specification language that is well suited for telecommunication systems and has been used extensively in the past few years for a wide variety of applications. Concepts from CSP, CCS, and data algebras have been combined in LOTOS, making it suitable for specifying protocols and services.

This paper describes our experience in using LOTOS to formally specify a subset of the General Packet Radio Service (GPRS) and the validation tools and techniques used. We have focused in this project on the service description stage 1 and stage 2 proposed by ETSI for GPRS. This work has faced many challenges mainly because of the following reasons:

- GPRS is a standard under design, hence it is still evolving.
- GPRS incorporates several features and while it includes some already known protocols, it also introduces some new ones.
- GPRS is a packet-switched mobile system. This implies complexity due to location management issues and packet delivery.

- The analysis of complex systems, such as GPRS, is constrained by the well-known problem of combinatorial state explosion.

We start by describing the approach used to capture the informal requirements and their translation into a formal specification. We then provide an overview of the GPRS system and give a background on the main LOTOS operators. We provide some details related to the LOTOS specification of GPRS before we focus on the validation activities. Finally we present the properties we have checked and we discuss in detail two design problems that we found.

2. From requirements to formal specification

Standards documents combine a mixture of text, tables, and visual notations such as *Message Sequence Charts* (MSCs) [3]. An iterative process is usually needed to go from the informal requirements to a high-level abstract and formal representation (formal specification). The evolving nature of standards, and the multitude of styles and notations used to describe them, add complexity to the process. One must not forget that standards tend to be quite unstable at their early stages and for this reason, the specification should be flexible enough to accommodate changes. Figure 1 illustrates the steps we used in this process. In the first step, *requirements capture*, we identify the *architectural requirements* which specify the various objects, processes, or entities of the system and their inter-relations and interfaces. We also extract the *behavioural requirements* which define the expected behaviour of the system. In the second step, *requirements synthesis,* we translate these captured requirements into a formal representation in LOTOS. The iterative nature of this process may reveal incompleteness, ambiguities, and inconsistencies in the specification. We can revisit the captured requirements and the original draft documents for several iterations (step 3).

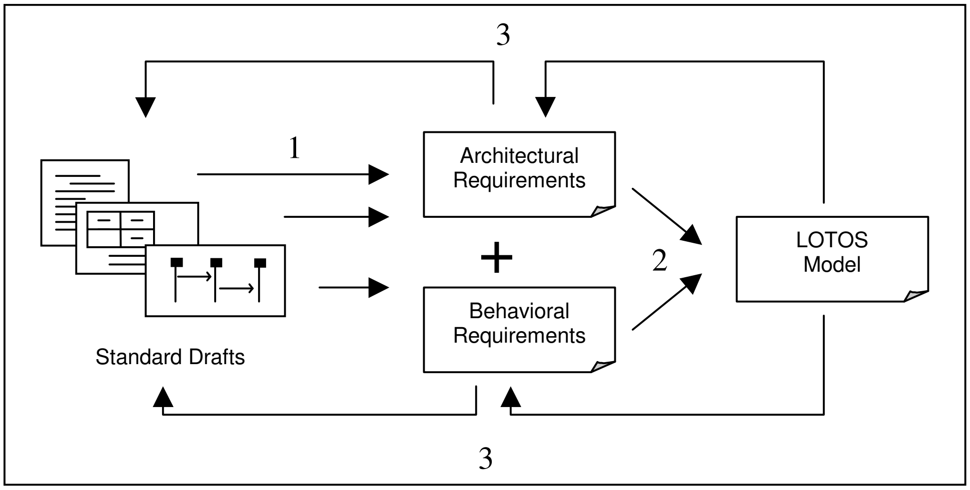

Figure 1. Requirements capture and modeling

Since it is executable, a LOTOS specification can be considered as a *model* or *prototype* of the system and can serve several purposes:

- It can be formally validated with respect to several properties.
- It can be used for system simulation.
- It can be used to generate scenarios (also called use cases) and test cases.
- It can be used as a guide for implementation.

We focus in this paper on the specification and validation steps of the requirements which are to be discussed later. Work related to the translation of informal requirements into a LOTOS specification can be found in [4].

3. Overview of GPRS

GSM (Global System for Mobile communications) [5] is a European standard for cellular communications developed by ETSI (European Telecommunications Standards Institute). GPRS is a set of new GSM bearer services that provide packet mode transmission within the GSM network and inter-works with external packet data networks [6]. It is a full digital system and is still an evolving standard that spans beyond telephony and circuit switched services. GPRS is a major activity in the phase 2+ of the GSM standard.

GPRS service subscribers will be able to send and receive data in a end-to-end packet transfer mode. GPRS Services are divided into two categories: Point-to-Point (PTP) and Point-to-Multipoint (PTM) services. Possible PTP services include data base access and information retrieval, the Internet, messaging and conversational services from user to user, credit card validation, etc. Examples of PTM services include unidirectional distribution of information such as news and weather reports. They also include conferencing services between multiple users.

3.1 Architecture of the GPRS network

GPRS introduces a new functional element to the GSM network (Figure 2): GSN (GPRS Support Node) which can be either a Serving-GSN (SGSN) or a Gateway-GSN (GGSN). This addition is necessary for the GSM network in order to support packet switched data services. We give below a summary of the main components of the GPRS network and their functions:

SGSN: responsibilities include maintaining the logical link with the Mobile Station (MS), forwarding incoming packets from the MS to the appropriate network nodes and vice versa, and authenticating access to GPRS services. Only one SGSN serves the MS in its service area.

GGSN: provides the interface to external Packet Data Networks (PDNs) and forwards packets destined for the MS to the SGSN that is serving it.

HLR: the Home Location Register is a database that contains subscriber's information. The subscriber's service profile and location information are stored in the HLR.

VLR: the Visitor Location Register is a database that stores temporary information for visiting subscribers.

MSC: the Mobile Switching Center is in charge of the telephony switching functions and authenticates access to circuit-switched services.

BTS: the Base Transceiver Station handles radio transmission and reception devices, including the antennas, and also all the radio interface signal processing.

BSC: the Base Station Controller manages the radio resources and controls handovers between cells. Several BTSs can be managed by one BSC.

Several interfaces have been introduced in GPRS to define entity-to-entity interactions. For instance, the *Gb* interface is required between the BSC and the SGSN. Two GSNs communicate through a *Gn* interface, and the SGSN communicates through the *Gr* interface with the HLR to send queries and to receive subscriber information. The *Gi* interface which connects a GGSN to a PDN was left open in the standard to accommodate implementation preferences while the Gs interface between the SGSN and the MSC/VLR was left optional.

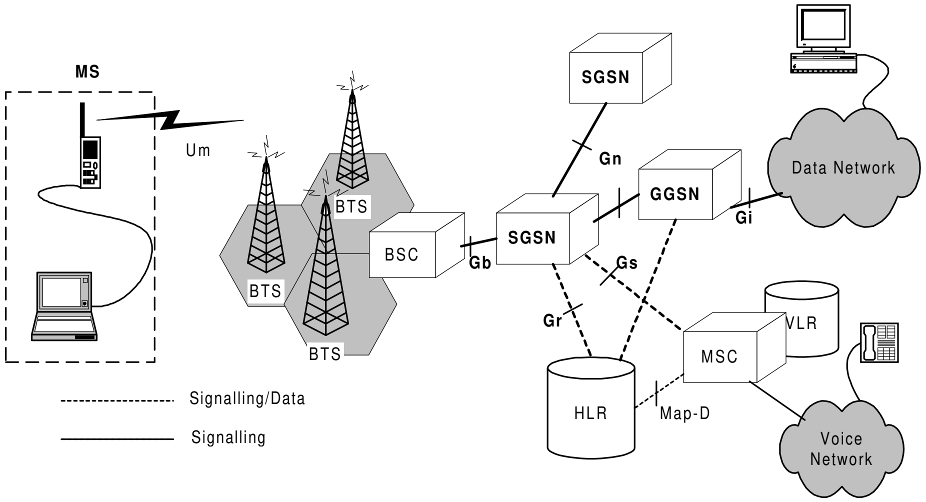

Figure 2. GPRS network architecture

The GPRS network is divided into several service areas assigned to different SGSNs. Each service area is composed of several Routeing Areas (RAs) which in turn form sets of cells.

3.2 Accessing the GPRS network

A MS can connect to the GPRS network by performing an *attach* procedure. The outcome is the establishment of a logical link between the MS and the SGSN and a mobility management context is created. Once this link is established, the MS can request to activate one or more Packet Data Protocol (PDP) contexts which specify the Packet Data Networks (PDNs) that it wants to connect to. In other words, it asks the SGSN to create routeing paths to the appropriate GGSNs. Once the PDP contexts are activated, the MS can send and receive data in an end-to-end packet transfer mode.

3.3 Mobility management states

The MS is defined to have three possible states: *Idle*, *Ready*, and *Standby* depending on the level of functionality it requires (Figure 3).

Idle State: a MS in this state is not traceable and can only receive multicast transmissions. The MS keeps track of cell changes locally. The MS needs to perform the *attach* procedure to connect to the GPRS network and become reachable.

Ready State: data is sent or received in this state and PDP contexts may also be activated and deactivated. Mobility management in this state happens at the cell level; the MS updates the SGSN when it changes cell. The MS may request a *detach* procedure in which case it moves to *Idle*. A timer monitors the *Ready* state and upon its expiry, the MS goes to the *Standby* state.

Standby State: a MS which has been inactive for a period of time is put in *Standby* state. The MS keeps track of the cell changes locally and informs the SGSN of RA changes only. It is possible in this state for the MS to activate PDP contexts establishing routeing contexts for data transmissions and receptions. The MS may wish to terminate the connection by requesting a GPRS *detach* procedure in which case it returns to *Idle*. A *Standby* timer also monitors the MS activity and causes it to go to *Idle* upon expiry.

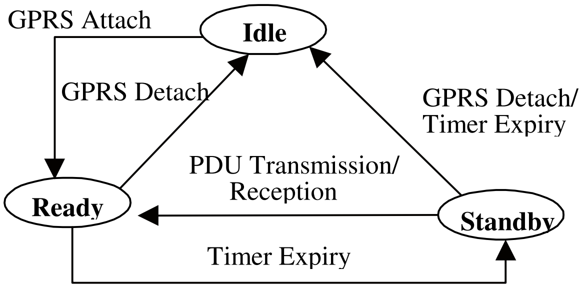

Figure 3. Mobility management state model

3.4 Location management in GPRS

Location management is the means by which the GPRS network keeps track of the MS position. Specific location management procedures are required depending on the current mobility management state of the MS. Since in GPRS the network is divided into smaller SGSN service areas containing several RAs, the term routeing update is used to refer to the location management procedures for GPRS. Geographically speaking, there are two types of procedures: *intra-SGSN* routeing update when the MS switches RAs within the same SGSN service area and *inter-SGSN* update to reflect a change of the SGSN service area.

4. LOTOS background

It is not possible in this paper to give a meaningful overview of the language LOTOS. Below, we list the main operators, however we realize that only readers familiar with the language will be able to follow the examples of the LOTOS code.

A LOTOS specification describes a system via a hierarchy of process definitions. A process performs internal unobservable actions, and interacts with its environment (other processes) via interaction points called *gates*. A process can combine actions and behaviour expressions by means of operators as follows:

a; B: the *action prefix* operator ";" means that an action or a gate **a** precedes the behaviour **B**.

B_1 [] B_2: the *choice* operator means that the process will behave as **B_1** or as **B_2** exclusively.

B_1 || B_2: the *full synchronization parallelism* operator means that **B_1** and **B_2** must synchronize on every action they offer.

B_1 ||| B_2: the *interleaving* operator expresses parallelism between **B_1** and **B_2** when no synchronization is required.

B_1 |[g_1,g_2,..g_n]| B_2: the *selective parallel* operator expresses parallelism between **B_1** and **B_2** when synchronization is only required on gates **g_1, g_2,..g_n**.

B_1 [> B_2: the *disable* operator means that any time during the execution of **B1**, **B_2** can take over, thus terminating **B_1**.

B_1 >> B_2: the *enable* operator means that **B_2** can be activated only after **B_1** terminates its execution successfully.

As mentioned in the introduction, LOTOS combines ideas from several preexisting formalisms. Value exchanges between processes are defined similarly to CSP [7]. Value offers are denoted by "!", and value acceptances are denoted by "?". For example, *g ! 3 ? y: int* denotes offering a 3 and at the same time accepting a value for *y* at gate *g*. LOTOS formal semantics is mainly based on CCS [8]. Hence LOTOS operators have a number of algebraic properties, and also have executable semantics based on inference rules. Being executable, a LOTOS specification provides a formal prototype, or model, of its object system. A number of LOTOS execution tools exist, and the ones used in this project are documented in Refs. [9][10][11][12].

5. Formal specification of GPRS

5.1 Scope of the specification

5.1.1 Logical functions. Several groupings of logical functions have been defined for GPRS. Our specification covers the following functions:

- ***Network access control***

  Registration: the association of a mobile to PDPs and addresses within the network.

- ***Logical Link Management***

  Logical link establishment: occurs when the MS attaches to the GPRS network.

  Logical link maintenance: controls the status and state changes of the logical link

  Logical link release: this function de-associates the MS-SGSN logical link.
- ***Packet Routeing and Transfer***

  Routeing: determines the network node to which a message should be forwarded.

  Encapsulation: adds address and control information to PDUs for packet routeing.

  Tunneling: transfers encapsulated PDUs between two end-points in the network.
- ***Mobility Management***

  Location management: a set of functions responsible for keeping track of the mobile station.

5.1.2 Protocol layers. A layered protocol structure is adopted for the transmission and signalling planes. The functions we specified are supported by several layers (shaded layers in Figure 4). The SNDCP (SubNetwork Dependent Convergence Protocol) serves as a mapping of the characteristics of IP/X.25 to the underlying network. Mobility management functionality is supported by the GMM (GPRS Mobility Management) and SM (Session Management) layers. The LLC (Logical Link Control) layer provides a logical link between the MS and the SGSN and manages reliable transmission while at the same time supporting point-to-point and point-to-multipoint addressing. The RLC (Radio Link Control), MAC (Medium Access Control), and GSM RF (Radio Frequency) layers control the radio link, the allocation of physical channels and radio frequency. LLC PDUs between the MS and the SGSN are relayed at the BSS. The BSSGP (Base Station System GPRS Protocol) layer handles routeing and QoS between the BSS (Base Station System) and the SGSN. The GTP (GPRS Tunneling Protocol) is the basis for tunneling signalling and user PDUS between the SGSN and GGSN. The remaining layers are already well known and defined protocols.

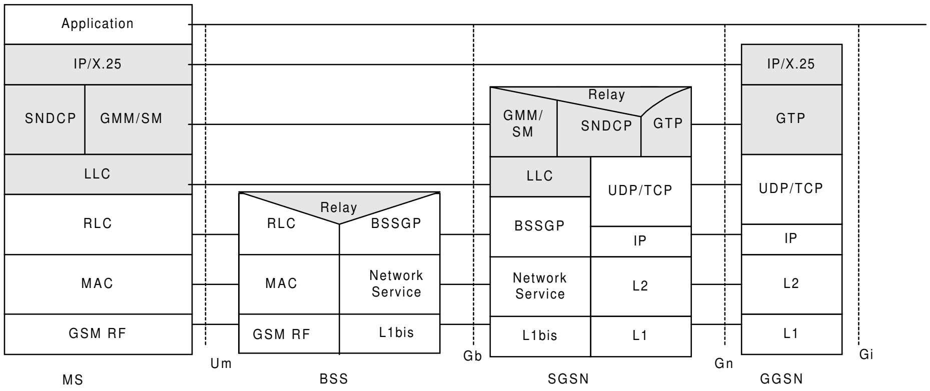

Figure 4. GPRS signalling and transmission planes from MS to GGSN

5.1.3 Model used and assumptions. We assume in our specification that the radio interface details such as the allocation of channels is somehow accomplished successfully. Since we are concerned mainly with the GPRS entities, we are more interested in the establishment of a logical link between the MS and the SGSN than in the medium used. For this reason, we chose to abstract from the details related to the radio interface in order to simplify the prototype and to focus on pure GPRS functionality. Functions related to the BTS and BSC are outside the scope of our specification. In our model shown in (Figure 5), we chose to adopt the following assumptions:

- The GPRS network is composed of four RAs. One RA is composed of two cells.
- There are one SGSN and one MSC/VLR for each pair of RAs.
- There is one HLR in the network.
- Two GGSNs serve as connections to the external networks. Both SGSNs can connect to either of them.

In a mobile network, the most interesting functions are usually related to mobility management (how to keep track of the mobile). In GPRS, complex location management occurs when the mobile switches SGSNs. By using this model, we are guaranteed to address this scenario and all the logical functions outlined above.

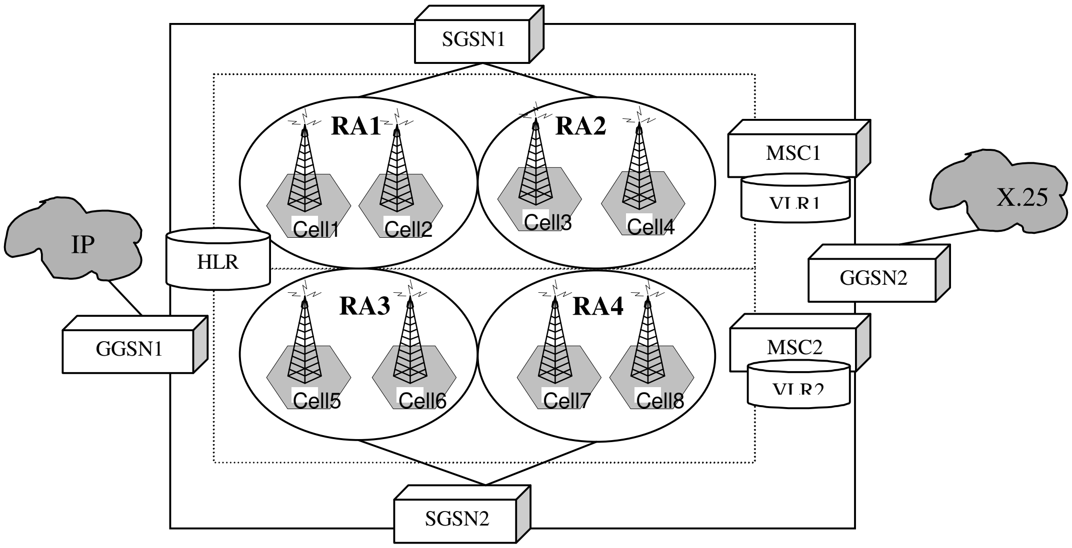

Figure 5. GPRS model used

5.2 LOTOS Specification Style and Architecture

Several LOTOS specification styles have been described in [13], and selected on the basis of their suitability to express design concerns. Choosing the right specification style is very important when constructing a LOTOS specification. In our work, we used the *resource-oriented* style, because it supports modularity and parallel structures. The term *resources* here refers to system entities with well defined interfaces to other entities. This choice is natural because it eases the mapping of a system's architecture into a LOTOS specification. System's components (resources) are described as LOTOS *processes* composed in parallel and interacting through *gates*. Other styles were used to specify detailed behaviour. The behaviour of the GPRS network entities (processes) is a collection of alternative sequences of interactions that depend on the mobility management states. The *state-oriented* style was a perfect choice to specify the behaviour of the individual processes. In some instances, the *monolithic* style was applied because events were presented as ordered collections of alternative sequences of interactions in branching time.

The GPRS system is seen as a LOTOS process interacting with the environment through the external gates *ms* and *ext_net*. These gates provide, respectively, the means to the user to initiate and simulate transactions of the mobile station and the external networks. They will be of particular use during the validation phase. By refining the GPRS process, we decompose it into three processes (Figure 6):

- **The_MS**: a set of interleaved Idle Mobile Stations (MS). The gate *Um_Gb* joins the *Um* and the *Gb* interfaces and carries messages between the MS and the SGSN. We needed this gate since we abstract from the radio link details.
- **GPRSNetwork**: an encapsulated process that is mapped to our geographical model described in Figure 5. In addition to the *Um_Gb* gate, messages are carried to and from the external networks via the *Gi* gate.
- **The_ExternalNetworks**: a set of external networks interleaving. An external network process is parameterized by a network type (IP or X.25) and communicates through the *Gi* and *ext_net* gates.

Due to the length of the specification, we will limit our explanation to the general structure of the GPRS network specification and give a high level description of the SGSN process.

The top level structure of the GPRS specification in LOTOS is shown in Figure 7. From the environment point of view, any message exchange occurring within the GPRS system is internal and cannot be observed. Only events that take place at gates *ms* and *ext_net* are observable. This explains the use of the *hide* operator within the GPRs process. The gates *Um_Gb*, *Gi*, *STmeout*, and *RTimeout* are therefore hidden to reflect their internal nature. The three processes *The_MS*, *GPRS_Network*, and *ExternalNetworks* run in parallel but have to synchronize on certain gates in order to communicate. This is expressed in LOTOS by using the selective parallel composition operator |[..]|.

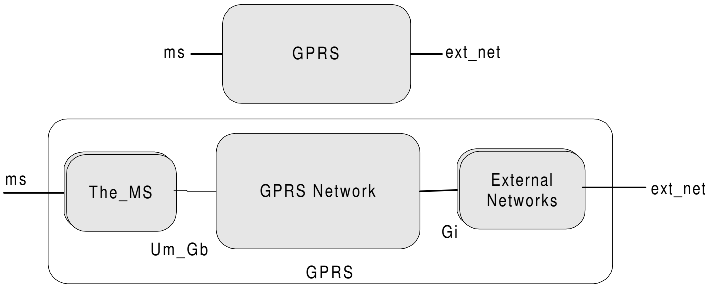

Figure 6. High-level specification architecture

We also need to add the two special gates *STimeout* and *RTimeout* to provide a mechanism for the SGSN and the MS to synchronize on *Standby* and *Ready* timers expiry, but explicit timing constraints are not modeled.

```
 specification GPRS_Spec[ms,ext_net]:noexit
        (*Data type definitions omitted*)
         behaviour
        GPRS[ms,ext_net]
         where
            process GPRS[ms,ext_net]:noexit:=
              hide Um_Gb,Gi,STimeout,RTimeout in
                ( The_MS[ms,Um_Gb,STimeout,RTimeout]
                      |[Um_Gb,STimeout,RTimeout]|
                  GPRSNetwork[Um_Gb,Gi,STimeout,RTimeout]
                 )
                |[Gi]|
              ExternalNetworks[ext_net,Gi]
        endproc (* GPRS *)
endspec (*GPRS_Spec*)
```

Figure 7. Top-level LOTOS specification of GPRS

It is easy to map our GPRS model of the GPRS network into a LOTOS process (Figure 8).

```
process GPRSNetwork[Um_Gb,Gi,STimeout,RTimeout]:noexit:=

        hide Gr,map_d,Gs,Gn,inter_sgsn in

            (* Some initilialization *)
            (*One HLR synchronized with two MSC/VLRs interleaving*)
            (HLR[Gr,map_d](InitSet,InitPDPSet)
              |[map_d]|
             (MSC_VLR[Gs,map_d](VLR(1),{} of MscVlrAssSet,0)
                |||
             MSC_VLR[Gs,map_d](VLR(2),{} of MscVlrAssSet,0) ))
                |[Gr,Gs]|
              (SGSN[Um_Gb,Gr,Gn,Gs,inter_sgsn,STimeout,RTimeout](SGSN(1),LA(1))
                  |[inter_sgsn]|      (*Two SGSNs synchronized*)
               SGSN[Um_Gb,Gr,Gn,Gs,inter_sgsn,STimeout,RTimeout](SGSN(2),LA(2)) )
                |[Gn]|                (*Two GGSNs interleaving*)
              (GGSN[Gi, Gn](GGSN(1))
                  |||
               GGSN[Gi, Gn](GGSN(2)) )

  endproc (* GPRSNetwork *)
```

Figure 8. Specification of the GPRS network process

By using the appropriate parallel operators, we can compose the various entities (processes) so that they reflect the architectural requirements. For instance, the GGSNs do not need to mutually synchronize on any action and therefore we use the interleaving operator (|||) to express their unsynchronized parallelism. On the other hand, they must interact with the SGSNs through the *Gn* interface. The selective parallel composition operator is appropriate in this case.

The SGSN process consists of several instances of *SGSNHandlers* synchronized with a process *SGSNManageLLC*. Data structures used for routeing and mobility management purposes are handled by process *SGSNContextsDBase*. A handler has been defined for each functionality performed by the SGSN such as handling mobile requests to attach to the network or routeing update requests. The handlers are interleaved, but since the *detach request* may occur at any time, the process *SGSNHandleDetach* has been composed with the disable operator ([>). By doing so, when a mobile user wishes to detach from the network, the request can disrupt the execution of the other handlers. Figure 9 illustrates the SGSN process architecture.

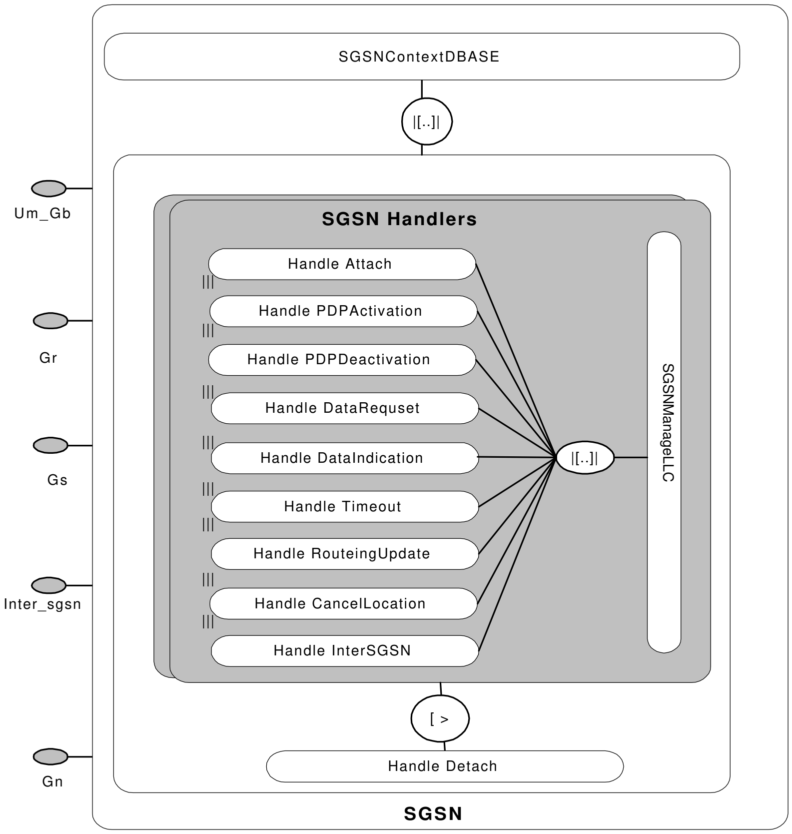

Figure 9. SGSN process architecture

6. Validation of the specification

Our methodology for the validation of the specification combines concepts of testing and model checking. We mean by testing the process of composing the specification in parallel with a test process which will force the specification through certain desired scenarios. This is done because it is impossible to execute all behaviour paths and different testing processes can be chosen to select the scenarios of interest. It is then possible to execute other validation activities such as model checking on the limited behaviour model narrowed by the test scenarios.

The steps involved in our validation methodology (Figure 10) are the following:

1. *Requirements capture and synthesis*: this step was described in Section 2 and its outcome is a formal specification of the requirements. In our case, this formal specification is expressed in LOTOS.

2. *Test definition:* we start by identifying the test objectives and then we derive the corresponding test scenarios. These scenarios will serve as a guide to the formal specification to perform specific execution sequences.

3. *Formulation*: some requirements can be expressed by means of a logical specification. The result is a set of properties that can be viewed as a partial correctness specification. In our work, we formulate several properties of the system using temporal logic.

4. *Composition:* the test scenarios derived in step 2 are composed with the formal specification to obtain test specifications. While the formal specification defines all the possible sequences of execution of the system, the test specifications are restricted to those sequences that are related to the test scenarios.

5. *Behaviour tree generation:* all possible paths in the test specifications are executed up to a certain depth yielding behaviour trees. These trees are graphical models representing the possible execution sequences dictated by the test scenarios.

6. *Test exploration*: all sequences of execution in the behaviour tree should normally be conformant to the test scenarios. However, erroneous sequences can be present indicating faulty behaviour. Most commonly, they show a situation where a specification deadlocks with a tester. In this step, we aim at finding such sequences.

7. *Model checking*: the properties formulated in temporal logic are checked against the behaviour trees of step 5. A formula that is not satisfied may reveal an erroneous behaviour. This step complements the test exploration activities because it addresses specific properties rather than scenarios.

8. *Analysis:* the tests and model checking results are then explored to identify the causes of the discovered problems. This step involves extraction of faulty traces of execution, and the generation of the corresponding MSCs for inspection.

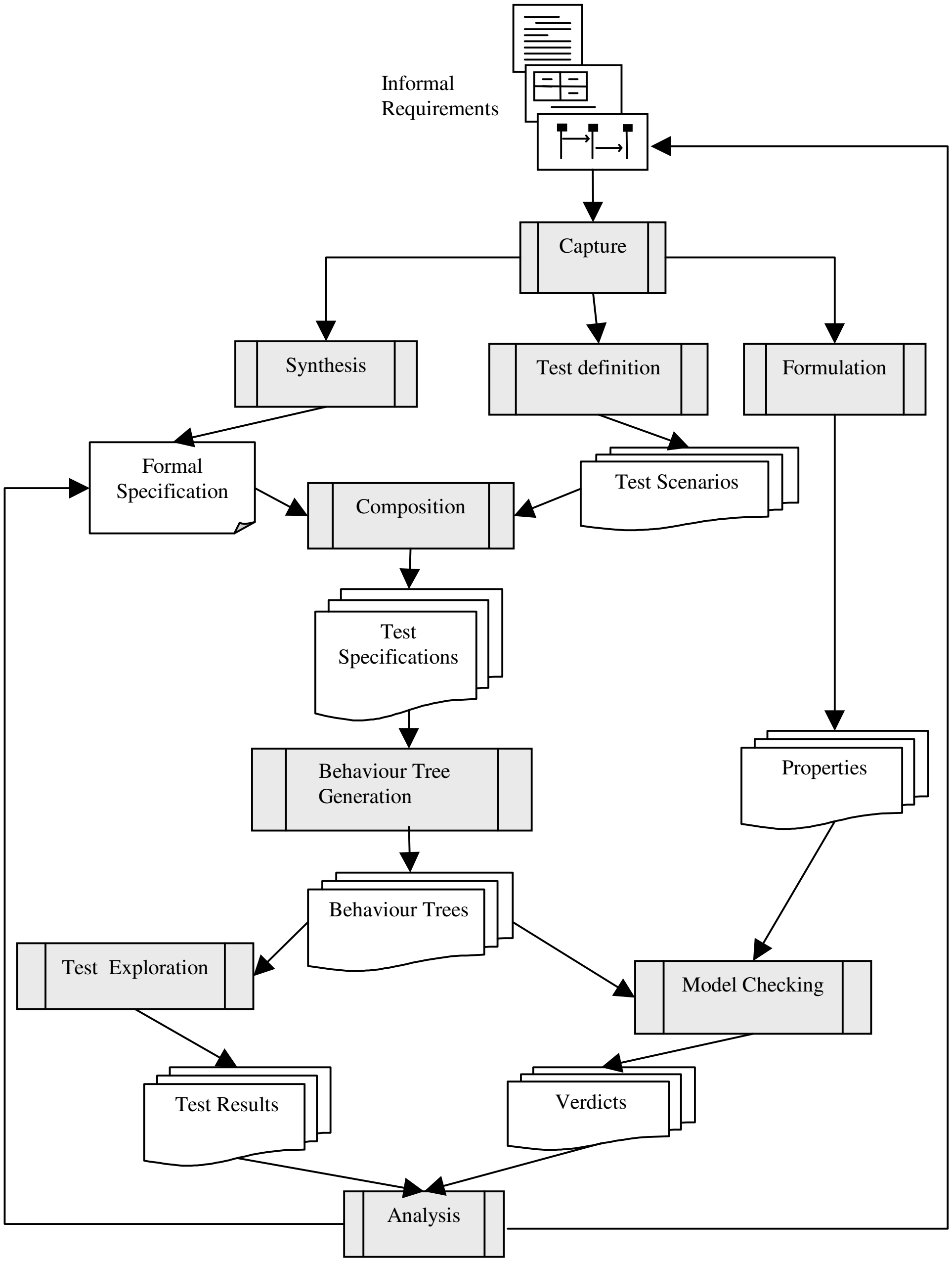

Figure 10. Validation methodology

6.1 Test definition

A test scenario describes possible behaviour sequences of the system and may reflect several interacting GPRS functions. We express these tests in terms of LOTOS processes. This approach follows the definition of *Testing Equivalence* [14]: tests are exercised by specifying a test process and synchronizing it with the behaviour under test. Our tests are characterized by two parts: an informal part consisting of their objective, a set of preambles and conditions, and a body reflecting the desired sequences of events. The LOTOS process represents the formal part. For example, in the preamble of the *intra-SGSN routeing update* test, we decide on some necessary information such as system configuration, the initial cell of the MS, the network connections, etc. To be able to test such a scenario, we need to attach the MS first, then trigger the routeing update procedure by forcing the MS to switch to another cell (cell 3 in this case), then detach the MS from the network. We reflect these actions in the body of the test and express them in the LOTOS process (Figure 11). We restrict the number of the *Ready* state timeouts to 1 to avoid the state explosion problem. We allow however *Standby* state timeouts to occur at any time by using the LOTOS disable operator ([>).

The special event “success” in the LOTOS test process is inserted to mark the end of every possible sequence of execution. It is used during the test exploration step to determine the existence of faulty sequences which are those not ending in “success”.

Note the structure of actions in this process. Because it is a test, all actions offer values only. Further, each action offers the gate name first (*ms*), followed by an identifier that identifies the mobile station (*imsi*), followed by on or more parameter values.

Test 4_1

Informal Description

Objective:
  Intra-SGSN Routeing Update

Preamble:
  1 MS with IMSI(1)
  MS in Cell 1
  NSAPI(0) associated with PDPAd(1), IP
  NSAPI(1) associated with PDPAd(1), X.25
  IP network managed by GGSN(1)
  X25 network managed by GGSN(2)

Body:
  GPRS Attach , Select Cell 3, and GPRS Detach

Conditions:
  1 Ready timeout may occur (restricted to 1 to avoid state explosion)
  StandBy timeouts may occur

Formal Description

LOTOS Process:

```
process TS4_1[ms,ext_net,success]:exit:=
 let imsi:IMSI = CreateIMSI(1) in  (* Intitialization*)
    ms ! imsi ! GPRSAttach; (* Initiate a GPRS Attach *)
     (ms ! imsi ! cellSelection ! Cell(3); (* Move to Cell 3 triggering intra-SGSN RA update *)
     ms !imsi ! RoutingUpdateAcc;
     ms ! imsi ! GPRSDetach;   (* Initiate a Detach *)
     ms ! imsi ! DetachAcc;
     success;
     exit)
      [>
     (ms ! imsi ! TimerExpAcc; (* Timeouts are possible *)
       success;
       exit )
 endproc
```

Figure 11. Test definition example

4 test suites were identified for our system with a total of 35 tests. The test suites are organized as follows:
Suite 1 (3 tests): Attach and detach procedures.

Suite 2 (8 tests): PDP context activation and deactivation procedures.
Suite 3 (13 tests): Data delivery (from MS to external network and vice versa).
Suite 4 (11 tests): Routeing update procedures and data delivery.

6.2 Composition

To compose a test process with the GPRS specification, we explored two different LOTOS tools. We first used LOLA [10] which performs the composition automatically. However, for some test processes containing interleaved sequences, LOLA was unable to handle the composition. We then resorted to RTL [12] to overcome this problem. The generation of the test specifications is done manually by synchronizing the test process and the GPRS specification using the LOTOS selective parallel operator (|[..]|).

6.3 Behaviour tree generation

The result of the execution of all paths of a LOTOS formal specification is a graphical model or representation describing the complete behaviour of the system it represents. This representation is called a Labeled Transition System (LTS) [8][15]. An LTS intuitively encodes the operational behaviour of a process. It contains the set of states the process may enter, and the set of actions the process may perform at each state. An LTS is a state transition machine where each edge is labeled by a LOTOS action.

For systems with infinite behaviour, a complete LTS cannot be generated. This problem is usually solved by specifying limits for the execution such as a maximum number of actions for each path. Another serious drawback is related to the well-known state explosion problem. In our system, the state explosion problem is due to two main reasons:

- Several processes run in an interleaved manner causing the number of possible actions and global states to increase very rapidly with time.
- The GPRS standard describes timers to monitor the mobility management state transitions. Timeouts are expressed in our system as events that can occur at any time. This multiplies further the number of actions and states to be explored.

We made some restrictions as to the number of timeouts allowed depending on the complexity of the tests to keep the state space tractable. Also, since we do not generate the LTS from the formal specification but rather from its composition with the tests, the size of our LTSs is controllable because they include only actions dictated by the test scenarios.

We used the RTL tool [12] to translate the LOTOS test specifications into LTSs. We then reduce the resulting LTSs with respect to some equivalence relations [15] using the Aldebaran tool [16].

6.4 Test exploration

A test is successful if all execution sequences of its composition with the behaviour terminate normally. We have seen (Figure 11) that the "success" event is inserted at the end of the test processes to indicate a successful termination. The presence of an execution sequence in the LTS that does not end with the "success" event indicates the possibility of a deadlock situation. Our task is then to find out if such abnormal terminations exist in the LTS.

In testing theory [14], the composition of a formal specification with a test leads to one of the following test results:

- *Must pass*: all the possible executions terminate successfully.
- *May pass*: some executions terminate successfully.
- *Reject*: none of the executions terminate successfully.

All the failing tests in our suites had a *May pass* result. This implied the existence of some abnormal sequences. To be able to inspect such sequences, we needed to extract them from the LTS. We used the Exhibitor tool of the CADP toolbox [10] for this purpose. Given an LTS, Exhibitor is capable of finding paths with certain given characteristics.

6.5 Model checking

Starting from the captured requirements, we identified a set of correctness properties (over twenty) to verify against our GPRS specification. To formulate these properties, we used a variation of the propositional μ-calculus [17] interpreted over the actions of the LTS. μ-calculus is sufficiently powerful to express safety and liveness properties. The formulas expressing the properties are therefore a combination of μ-calculus syntax and LOTOS actions. We used the model checker of the CADP toolbox [10] to accomplish this step.

6.5.1 Examples of correctness properties. We do not present the syntax and semantics of μ-calculus here, and we only show one property expressed in its precise syntax. We list in natural language a subset of the properties verified to give an idea of the type of properties we addressed.

1. *When in Standby, the MS is not allowed to move to Idle unless a detach or Standby timer timeout occur.*

   Formulation: to express this property, we need to check that for every path in the LTS starting with a *Standby* state action "*state !IMSI1 !STANDBY*", an *Idle* state action "*state !IMSI1 !IDLE*" is observed only after a confirmation of detach "*ms !IMSI1 !DetachAcc*" or a timer expiry "*ms !IMSI1 !TimerExpAcc".*

```
ALL (
    ["state !IMSI1 !STANDBY"]
    SU (["state !IMSI1 !IDLE"]F)
      (<"ms !IMSI1 !DetachAcc”>T or <"ms !IMSI1 !TimerExpAcc”>T)
     )
```

2. *Data transfers (PTP) to and from the MS are not possible during Idle.*
3. *After a successful attach to the network, it should be possible to detach.*
4. *The MS cannot receive or send PTP data in Standby.*
5. *A PDP context shall be activated before the MS can be paged, or can send, or receive data.*
6. *During an inter-SGSN routeing update, the new SGSN should request the MS context from the old SGSN.*

6.6 Analysis and results

Whether we use testing processes or model checking, the difficult task remains to find the causes of the abnormal sequences or failing properties. In case of testing, inspecting the traces of execution can be tedious and time consuming. Model checking on the other hand requires an elaborate diagnostics system to explain why a property was not satisfied by the system. To facilitate our task, we relied heavily on automatic generation of MSCs from traces of executions. Faulty sequences could then be inspected visually in a more efficient way. MSCs are also generated for correct sequences of execution and compared with the original MSCs defined in the standard documents. Alternative solutions to a specific scenario can be provided in this way. Figure 12 shows an example MSC that was generated for a successful *attach* procedure. We chose to show this simple MSC because it is impossible to provide a more complex trace of execution without compromising its readability.

Several specification problems were detected using our validation techniques. The two main ones are described below:

1. *Mobility Management state conflicts*: these problems are due to timer expiry during the reception of data destined to the MS by the SGSN. The SGSN and the MS have to agree on the mobility management states. In our analysis, we identified a case where the SGSN and the MS engaged in a conflict. While the MS switched to *Standby*, the SGSN was still in the *Ready* state. Packets arriving to the SGSN could not be forwarded properly to the MS. This problem was addressed in later versions of the standard by adding recovery mechanisms from inconsistent mobility management states in the MS and the SGSN.

2. *Routeing updates and data delivery conflicts*: these conflicts occur when incoming packets destined to the MS arrive at the SGSN during a routeing update procedure. During an intra-SGSN update, the SGSN needs to forward the data to the new RA. On the other hand, the old SGSN needs to forward the data to the new SGSN during an inter-SGSN update. When a routeing update procedure is in progress, the data gets lost. This issue was addressed later in

the standard by adding a buffering mechanism that allows the SGSN to hold the data until it completes successfully the routeing update procedure, and then forward it to the right destination.

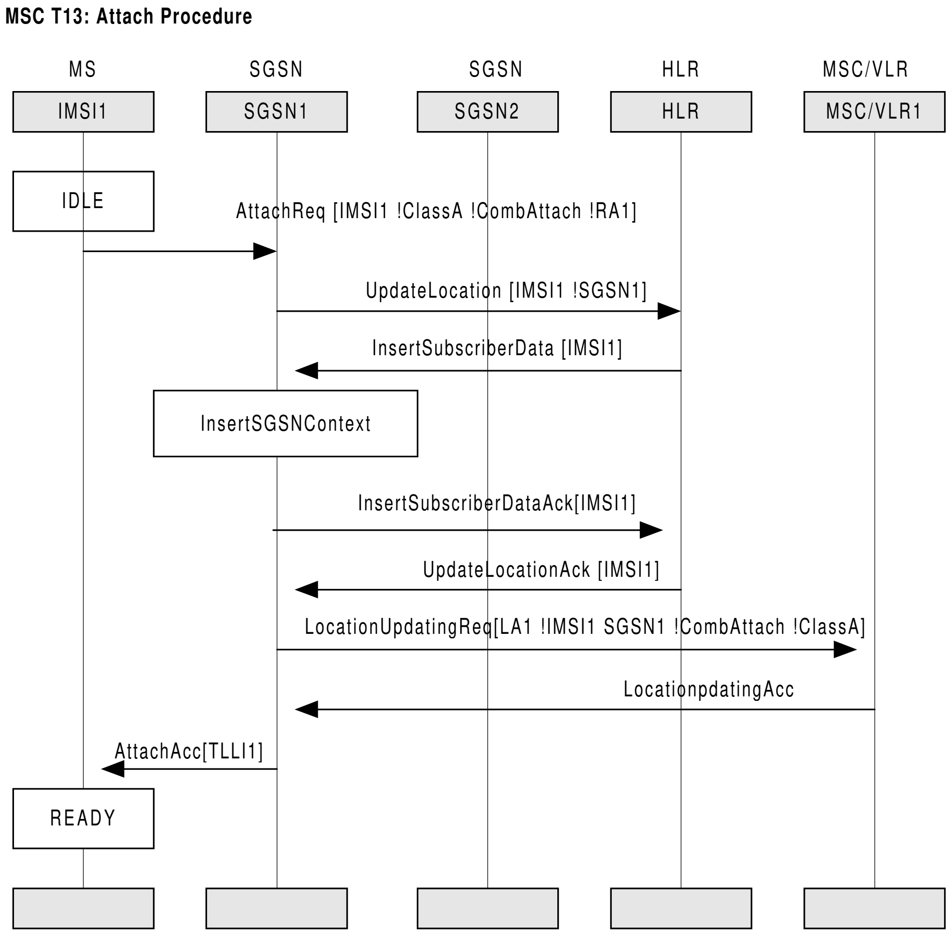

Figure 12. Successful attach procedure (MSC)

7. Conclusion

In the mobile data communication area, market forces oblige standard committees to develop new standards rapidly. These standards include complex protocols that should be checked for correctness before they are implemented, otherwise problems will result, leading to high development costs, customer dissatisfaction or even the failure of major systems that use the standards.

In this paper, we have presented an experience in the use of formal methods for the verification of crucial properties of a forthcoming standard. Although the methods used are based on existing ideas, the application to this complex system has required a lot of ingenuity, from developing a suitable formal model for the system, which was continuously being modified, to making the tools work towards our goals.

At least two significant design errors were found. Each one of them, if not addressed, would have led to faulty functioning of the system, possibly involving loss of data or of connectivity to the mobile user.

This experience confirms the usefulness of formal methods in the design of dependable critical systems.

Acknowledgment

This research was funded by grants of Motorola Canada, and of the Natural Sciences and Engineering Research Council of Canada. We would like to thank Bernard Stepien for his early contribution to this project, and the members of the LOTOS group for their helpful comments and discussions.

References

1. ISO, Information Processing Systems, Open Systems Interconnection, *LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*, IS 8807 (E. Brinksma, ed.), 1988.
2. L. Logrippo, M. Faci, M. Haj-Hussein. An Introduction to LOTOS: Learning by Examples. Computer Networks and ISDN System, 23 (1992), 325-342. (errata in 25 (1992) 99-100).
3. ITU, Recommendation Z. 120: Message Sequence Charts (MSC), Geneva (1996).
4. L. Logrippo and R. Tuok. Formal Specification and Use Case Generation for a Mobile Telephony System. Computer Networks and ISDN Systems, 30 (1998), 1045-1063.
5. M. Mouly and M. B. Pautet. The GSM System for Mobile Communications, Palaiseau, France, 1992.
6. ETSI, Digital cellular telecommunications system (Phase 2+): General Packet Radio Service, GSM 02.60 Service Description Stage 1, version 5.1.0, October 1997 and GSM 03.60 Service Description Stage 2, version 5.2.0, December 1997.
7. C.A.R Hoare. Communicating Sequential Processes. Prentice-Hall, 1985.
8. R. Milner, A Calculus of Communicating Systems, Lecture Notes in Computer Science, Vol.92, Springer-Verlag, 1980.
9. B. Ghribi. And L. Logrippo. A Validation Environment for LOTOS. In: Danth