A Case Study in Formal Methods: Specification and Validation of the OM/RR Protocol

Tim Willemse[1], Jan Tretmans[2], and Arjen Klomp[3]

[1] Eindhoven University of Technology, Formal Methods Group,
Department of Mathematics and Computing Science, P.O.Box 513, 5600 MB Eindhoven,
The Netherlands, timw@win.tue.nl
[2] University of Twente, Formal Methods and Tools group, Faculty of Computing Science, P.O.Box 217,
7500 AE Enschede, The Netherlands, tretmans@cs.utwente.nl
[3] CMG Den Haag B.V. Division Advanced Technology, P.O.Box 187, 2501 CD The Hague,
The Netherlands, arjen.klomp@cmg.nl †

Abstract. This paper reports on the results of the application of formal methods in the development of an industrial, mission-critical system, called the *Operator Support System*. A critical communication protocol of this system, the *OM/RR Protocol*, and its corresponding service were formalised using the formal specification language LOTOS. The resulting specifications have been validated using the tool set LITE and models of the specifications, obtained by making abstractions, have been verified using the tool EUCALYPTUS. Whereas the use of formal methods is usually motivated by their ability to allow for unambiguous and precise system descriptions amenable to mathematical reasoning, it turned out that in this project most benefits were obtained by the sheer process of *formalising* the informal protocol description, revealing many omissions and ambiguities. The results and experiences obtained during formalisation, validation, abstraction and verification are discussed on a non-formal basis in this paper.

Keywords & Phrases: application of formal methods, industrial case study, LOTOS, communication protocol.

1 Introduction

Formal methods have always been envisioned to be applied to systems that require unambiguous, mathematically precise descriptions for their correctness. The use of formal methods in the industry has been prophesied since their very dawn, however, reality learns that their acceptance is still not very wide-spread. One way of increasing the acceptance is by applying formal methods in case studies of real industrial systems. This paper reports on such a case study: the *OM/RR protocol*. The project was conducted between 1997 and 1998.

The *OM/RR protocol*, which is a short-hand for *Object Management/Request Response protocol*, is a communication protocol used in the *Operator Support System* (OSS). OSS is a complex, *mission-critical system* which is being developed to integrate different *Motor-way Management Systems* (MMSs). CMG Den Haag B.V. takes part in this development which is commissioned by the Traffic and Transportation Department of the Dutch Ministry of Transport, Public Works and Water Management.

In the past years, various MMSs have been developed, such as fog detection systems, congestion detection systems and variable message signs for controlling traffic. For historical reasons, many MMSs have their own specification and implementation and they cannot interact. The control of an MMS takes place at an Operating Centre. The current situation in The Netherlands is that

† Current affiliation: CMG Eindhoven B.V. Sector Trade, Transport & Industry
Luchthavenweg 57, P.O.Box 7089, 5605 JB Eindhoven, The Netherlands

there are several Operating Centres throughout the country. Each Operating Centre has its own domain, consisting of several MMSs. Since these domains are piecewise disjoint, this severely limits the — nowadays much needed — cooperation between different Operating Centres.

In recent years this awareness has grown, leading to the description of the OSS. This system is intended to integrate the independent Operating Centres and MMSs, and thereby to increase the power of the system regarded as a whole. The idea is to enable one Operating Centre to take over duties from another Operating Centre, or to perform tasks that at some point go beyond the Operating Centre's own domain, thus offering maximal flexibility in regulating and controlling traffic.

The cooperation between the Operating Centres is supported by communication and authorisation protocols. The communications protocol is intended to allow both communication between Operating Centres and between an Operating Centre on the one hand and an MMS on the other hand. The authorisation protocol is intended to restrict all possible communications allowed by the communications protocol, avoiding possibly chaotic situations, in which control over MMSs can be lost. Both protocols display complex behaviour and are critical to the well-functioning of OSS, so they are sources of potential hazards to the functioning of the OSS.

The main motivation for CMG to incorporate formal methods in the development of the OSS was the mission critical aspect of the OSS. Moreover, CMG had positive experiences with the use of formal methods in another mission critical system: the BOS system, the decision support system which controls and operates the storm surge barrier in the Nieuwe Waterweg near Rotterdam – a movable dam which should protect Rotterdam from being flooded [10, 20].

The main goals of applying formal methods to the OM/RR protocol were to check its feasibility and suitability for OSS, and to judge the completeness, preciseness and consistency of its (informal) specification. It was acknowledged that, based on the documentation available for the OSS, no firm conclusions about these facts could be drawn. It was expected that formal methods could help in giving an answer to these questions. A secondary goal for CMG was to increase their knowledge of and their experience with formal methods and associated tools.

In this paper, we describe the successful application of formal methods to the analysis of the OM/RR protocol – successful at least from the formal methods point of view, not from the OM/RR protocol point of view, as will be seen. By carefully translating the informal documentation for this protocol, taking into account the generally believed functionality of this protocol, a formal specification is constructed. This process allows one to determine omissions, as well as ambiguities in the informal documentation. Simulation of the formal specification is used to validate the formal specification against informal specifications and design requirements. Confidence in the correctness of the formal specification is increased by showing proper, formal relations between *models* of the formal service specification and *models* of the protocol specifications.

The concepts of *service* and *protocol*, used in this project, are defined in [21] and stem from the context of the OSI Reference Model [16]. It is illustrated in Fig. 1. A communication service is provided by a *service provider*, which is considered as a blackbox. The *service users* communicate with the service provider via shared interaction points, so-called *Service Access Points* (SAPs). The layer-n *service specification* defines the possible communications between the layer-n service users by using the layer-n service provider. The layer-n *protocol specification* describes how the layer n-protocol entities should communicate via the $n-1$ service provider to establish the n-service.

The rest of this paper is structured as follows. Section 2 discusses the communications protocol: its basic, informal documentation, its ambiguities, its omissions and its characteristics, and the choices that have been made in the process of formalising specifications are sketched. The validation and verification aspects of the formal specifications are the topic of Section 3, and concluding remarks and an evaluation of the use of formal methods in this project are discussed in Section 4. Full technical details of this project can be found in [23].

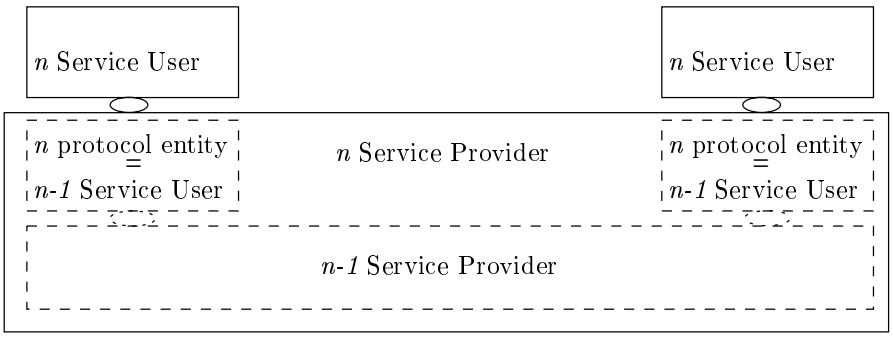

Fig. 1. *The Concept of Service*

2 The OM/RR Protocol

This Section describes the informal and formal OM/RR protocol: the documentation for the OM/RR protocol is discussed in Section 2.1. Subsequently, Section 2.2 discusses some of the problems and omissions of this document. Then Section 2.3 describes briefly the process of formalising the documentation for the OM/RR protocol.

2.1 The OM/RR Documentation

The starting point for the formalisation was the description of the OM/RR protocol in [7], which was developed outside the current project. According to this document, the OM/RR protocol is loosely modelled after the OSI CMIS [5], the OSI CMIP [6], some parts of Systems Management [1] and the OSI Remote Operations [4] standards.

The document [7] is, including appendices, a little over 75 pages, describing the OM/RR protocol and Association Management. Several appendices are added to explain short-hand notations and conventions, taking up 15 pages of the document. A few pages are dedicated to a more global, informal, description of the OM/RR protocol and its relation with the OSS, creating a context for the OM/RR protocol.

The contents of [7] consists mainly of information about data, described in the notation ASN.1 [19, 3]. Using this notation, the *service primitives* (i.e. primitives for communication between service users and a service provider) and the *protocol data units* (i.e. units of data, handled by a protocol entity) of the OM protocol entities and the RR protocol entities are specified. Although ASN.1 allows one to write data in a concise and platform independent way, it does not allow for specifying dynamic behaviour. The OSI standards CMIS and CMIP also do not encompass any dynamic behaviour, as they leave this part up to the users of these standards. Other issues, discussed in [7] deal with the decomposition of the OM/RR protocol in layers in an OSI/RM manner (see Fig. 2).

2.2 Omissions

During careful, thorough analysis and study of the informal OM/RR document with the intent of developing a formal description a lot of incompleteness, impreciseness, ambiguity and inconsistency was discovered.

Although OSI/RM-like layering of Fig. 2 is visually very compelling, additional information is required to understand the interaction mechanisms between the different protocol entities and layers. However, the functionalities of each layer are discussed only informally, ambiguously, or not revealing the inter-dependencies between the protocol entities. This lack of information about the entanglement of these protocol entities is regarded as a major omission.

Another major omission is the absence of descriptions for the dynamic behaviour (e.g. a service specification), except for a transition diagram described in one appendix and a scenario describing

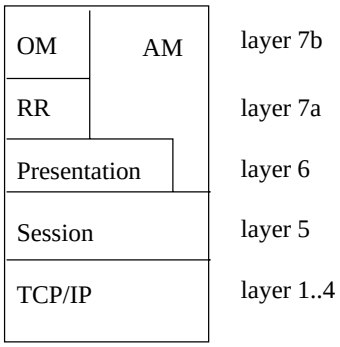

Fig. 2. *The protocol stack*

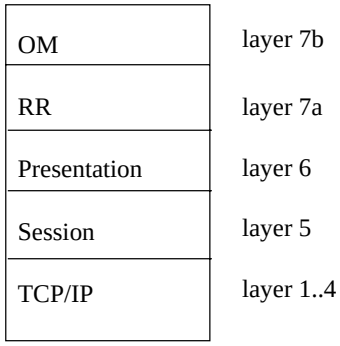

Fig. 3. *The alternative protocol stack*

aspects of communication for association management. The status of this diagram and the scenario, however, is not explained. Service specifications are vital for users of a protocol, since the service specification accurately describes the behaviour to be expected by a service user. The (formal or informal) protocol specifications are necessary for guiding a programmer to a correct program. Absence of both means that the documentation is not complete.

The dynamic behaviour is always described based on certain *design requirements*. Using these requirements, it should be possible to design a service specification and protocol specifications. Most design requirements, however, are not described in [7]. The design requirements that are documented in [7] are mostly concerned with performance aspects instead of functionality. As such, the documentation for the OM/RR protocol fails to meet its purpose.

Conversations with the developers revealed that they regarded the specifications of the service primitives and the protocol data units as a sufficient basis for implementing the OM/RR protocol. Specifying dynamic behaviour was regarded to be a non-issue, since it was, as they believed, "trivial". Within CMG, though, the opposite was believed. This feeling was confirmed, since the trajectory of formalisation revealed that various issues proved more complex than the developers had expected.

At this point in the project, one of the goals of the formalisation process had, in fact, already been achieved: the OM/RR protocol document [7] was not stable, complete and precise enough to form an unambiguous basis for further development of conforming implementations.

2.3 Formalisation of the OM/RR Protocol

Analysis of the informal OM/RR protocol document [7] revealed that this document is not suitable to be considered a basis for formalising the protocol, see Section 2.2. Despite this fact, we decided to continue the formalisation process to see to what extent the other goal – checking the feasibility and suitability of the design of the OM/RR protocol for OSS – could be met. An attempt has been made to design formal specifications for the OM/RR protocol. Using the few design requirements, documented in [7], information obtained in conversations with the developers of [7] and information within CMG from people working on the OSS project, design requirements have been formulated. These design requirements then served as the basis for formalising the OM/RR protocol. The focus in the formalisation process has been on the dynamic behaviour, since the data was already fully specified using ASN.1. It should be noted that now we not only formalised the existing protocol, but also were actually designing parts of it.

Subsequently, in the process of formalisation, a choice for a suitable formalism for describing the specifications had to be made. This choice was made by considering two criteria to which the formalism should adhere:

1. The formalism should allow for a concise and unambiguous specification of dynamic behaviour;

2. The formalism should be supported by tools able to analyse, validate and verify descriptions in this formalism.

Even though a number of formalisms adhere to these requirements, a slight preference for a process algebraic formalism led to LOTOS [8, 2]. The tool support used in this project for LOTOS consists of the tool-sets LITE [12] and EUCALYPTUS [14, 13] (various 1998-beta-versions of CADP 97b "Liège").

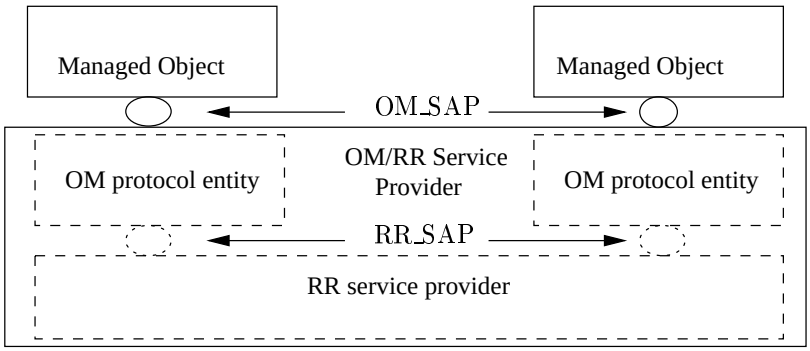

Fig. 4. *The Relation between the OM/RR Service, the OM Protocol and the RR Service*

The service specification of the OM/RR system – actually the OM service – is based on the design requirements formulated for OM/RR. Subsequent decomposition of this service specification leads to a lower-level service specification and a protocol specification. Visually, the decomposition is expressed in Fig. 4. This decomposition also has its impact on the OM/RR protocol stack (see Fig. 3 on page 4). The connection between the generic service/protocol structure in Fig. 1 and Fig. 4 is easily made. The service the OM/RR protocol, i.e. the *OM/RR Service Provider*, provides to its service users, i.e. *Managed Objects*, is described in LOTOS. The decomposition of the OM/RR Service Provider leads to specifications for the *OM Protocol Entities* and the *RR Service Provider*. These are also described in LOTOS. The correctness of this decomposition step can now easily be expressed in LOTOS as follows, where `OM_RR_Service_Provider[OM_SAP]`, `OM_Protocol_Entity[OM_SAP,RR_SAP]`, `RR_Service_Provider[RR_SAP]` are LOTOS processes for the OM/RR Service Provider, OM Protocol Entity and RR Service Provider, respectively; `OM_SAP` and `RR_SAP` are the Service Access Points of the OM layer and RR layer, respectively; and $\approx$ is a suitable semantic relation between processes.

```
OM_RR_Service_Provider[OM_SAP]  ≈                                    (2.1)
   HIDE RR_SAP IN    ((    OM_Protocol_Entity[OM_SAP,RR_SAP]
                        ||| OM_Protocol_Entity[OM_SAP,RR_SAP] )
                    |[RR_SAP]|  RR_Service_Provider[RR_SAP] )
```

Basically, the service specification of the OM/RR protocol describes communications between the service users of the OM/RR protocol, which are according to [7] *Managed Objects*. The service users impose, by definition, no restrictions on the OM/RR protocol. Managed Objects communicate with one another via *associations*, which are considered to be binary relations. Two Managed Objects connected with each other via an association are assigned *roles* based on the initiative in establishing the association. One takes the role of a *Manager* and one takes the role of an *Agent*. These roles have their impact on the set of service primitives a Managed Object is allowed to use in an association with another Managed Object. The order in which service primitives are used to establish, communicate and release an association, is described in LOTOS. The view a Managed Object has on an association is described by means of a state-transition diagram (see Fig. 5). This state-transition diagram served as a framework for writing the LOTOS specifications.

The specification of the OM protocol entities describes formally how the OM/RR service specification can be met, using the service specification of the RR protocol. The service specification of the RR protocol describes a connection-less service, offering reliable data transmissions between two service users. It is not clear to us yet, and in [7] it is also not explained, why the developers used a TCP/IP network below the RR layer (see Fig. 2 and 3).

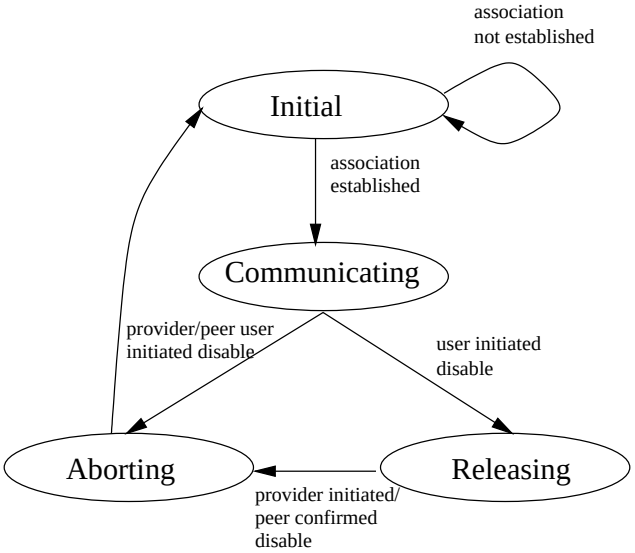

Fig. 5. *A Managed Object's view on an Association*

Specification	Lines of LOTOS code
OM/RR service specification	1000
OM/RR protocol specification	1600

Table 1. *Lines of* LOTOS *Code*

To give an indication into the complexity of the final specifications, an approximation of the number of lines of LOTOS code are mentioned in Table 1. In these specifications data not affecting the dynamic behaviour (e.g. data for the service users), is abstracted from. Clearly, the OM/RR service specification is specified more easily than the OM/RR protocol specification, which is an argument in favour of using the concept of service as a step in protocol development.

3 Validation, Verification and Analysis

A part of the objectives in our project has been to reveal the (lack of) information contained in documents concerning the OM/RR communications protocol. This part has been discussed in Section 2. Another objective has been to show how formal methods can be applied to write unambiguous specifications in an industrial project. Obtaining confidence in such specifications is also assisted by formal methods. The formal specifications written for the OM/RR protocol have been validated and verified using the tool-sets LITE [12] and EUCALYPTUS [14, 13]. These validation and verification efforts are discussed in the subsequent sections.

3.1 Validation and Analysis using LITE

The validation and analysis of the formal specifications of the communication protocol has been tackled in different stages. The first stage consisted of checking the syntax and static semantics of the LOTOS-specifications. For this, the syntax and static-semantic analyser of LITE were used. Although necessary for performing subsequent analysis, fixing the problems discovered in this stage only provided more insight into the description language rather than in the specification itself. Given the amount of errors that were reported in this stage, one may consider the FDT LOTOS either unnatural or hard to learn. The main problems encountered in this stage were of the nature of functionality of a process (i.e. `exit` versus. `noexit`), which at first seem very counter-intuitive.

The second stage involved checking the dynamic behaviour of the specifications. The design requirements and the state transition diagram (see Fig. 5) have been used as a guideline in this stage. The LOTOS simulator (animator) SMILE, which is part of the tool-set LITE, has been used for this. At first, a simulation based approach was taken, using SMILE to perform single stepping through the specifications. Single stepping is a simulation in which actions are presented that can occur *immediately after* choosing an action that has to occur. This analysis, however, is troubled by the fact that the overall behaviour of the system is "contaminated" with internal actions that are present in the specification. To overcome this problem, one can look at the *action prefix* to make an educated guess about which action leads to the desired observable action. Although in theory this can be used to perform some thorough simulations, it is also a very cumbersome method to use.

Another option that has subsequently been taken is by making use of test cases, in which certain behaviour of the system is expressed, i.e. so called *may* test cases [11]. In contrast to *must* test cases, in which the test case must specify behaviour which is *always* possible, may test cases specify only behaviour which *in some cases* is possible. Test cases control the behaviour of the specification into the direction expressed in the test cases. In this way (un)expected behaviour can be checked. The construction of the test cases that have been used, was guided by the structure of the state-transition diagram of Fig. 5. All test cases tested traces that started in the *Initial* state, and specified different routes back to the *Initial* state again.

Although the execution of test cases usually leads to a more restricted dynamic behaviour, in our case much of the behaviour was still blurred by the internal actions. Yet, many cases of faulty behaviour were revealed using the test cases. Several rounds of improvement and re-testing were necessary to mend the errors discovered at this stage. Note that in many cases, the translation of the ideas expressed in the natural language to the formal LOTOS-descriptions was incorrect, instead of the basic intuition behind these ideas.

By using the static analysers and the simulator SMILE from the tool-set LITE the confidence in the correctness of the LOTOS specifications was increased. However, the possibilities for further validation and analysis using tools from LITE are limited. This means that, in order to perform, for instance, checks for absence of deadlock, or establish the relation expressed by equation (2.1) formally, other tool-sets have to be used. The subsequent sections will go into more detail about the additional analysis and validation.

3.2 Validation, Verification and Analysis using EUCALYPTUS

The tool-set EUCALYPTUS [14, 13] is a collection of several tools for analysing and verifying LOTOS descriptions. However, the tools that come with this tool-set all have a major drawback in that they pose restrictions on the LOTOS grammar: not the full set of LOTOS operators is supported. The positive side is that these restrictions allow for more powerful tools, such as checking for deadlock, livelock, (bi)similarity and preorder relations. These problems are in general not computable for the full grammar of LOTOS. The use of this tool-set on the complete LOTOS specification as it is, however, is not possible due to this lack of support for the complete LOTOS grammar. In order to use the tools, simplifications and abstractions have to be made: *models* have to be constructed from the specifications. These models should then focus on some part of the vital behaviour of the

communication protocol. A second reason for developing models from the complete specifications is complexity. Although, with the restrictions on the LOTOS grammar the above mentioned checking problems are in principle computable, the size of the models, e.g. the number of system states, may prevent their effective computation by EUCALYPTUS. Also for this reason simplifications and abstractions are needed.

Although the use of models in the industry is fully accepted, various implications must be kept in mind. A model is a simplification of a real system, and as such it is often impossible to fully analyse a system. Also, the construction of a model can, on the one hand, introduce errors not present in the original specification, while, on the other hand, it may hide errors that are present in the original specification.

Models for the OM/RR protocol were developed from the formal specifications by abstraction and simplification. These models, and not the complete formal specifications, were the subject of verification with EUCALYPTUS. The justification for their development is given in Section 3.3, while Section 3.4 gives a brief documentation of the results obtained in the validation, verification and analysis of the models.

3.3 From Specifications to Models

The LOTOS constructs that are not supported by tools of the tool-set EUCALYPTUS are the following: no process recursion is allowed on the left and right hand part of the parallel-operator `|[..]|`, nor on the left hand part of the enable-operator `>>` and the disable operator `[>`. Furthermore, data type definitions are restricted to define only finite and enumerable sorts, instead of possibly infinite sorts. Most of these problems can be overcome by considering only a subset of the behaviour of the specifications. Design requirements in the case of the OM/RR protocol that led to a LOTOS description using unsupported LOTOS constructs are the following:

- An unbounded number of concurrent associations should be supported.
- An unbounded number of messages can be in transit in the communicating state at any moment in time.
- Each association can have infinitely many *sessions*, where a session is the period between and including the successful establishment and release of an association.

In the LOTOS specification, each association is described as an independent entity, not affecting (and aware of) the other associations. The unbounded number of associations is specified using process recursion combined with an interleaving operator (see Equation 3.2). Process `associations` represents the dynamic behaviour of *all* associations. Each particular association, specified by process `association`, is identified by an element of the (possibly infinite) set `AIdSet`.

```
PROCESS associations[OM_SAP] ( A : AIdSet ) =                          (3.2)
    CHOICE aa : AId [aa isin A] ->
    (   association[OM_SAP] (aa)
    ||| associations[OM_SAP] (remove (aa,A) ) )
ENCPROC (* associations *)
```

The use of the interleaving operator means there is no synchronisation on service access points between different associations, hence they are specified as being truly independent entities; therefore, it is fair to assume that considering only *one* association does not affect the validity of results for an unbounded number of associations. Thus, the model considers only process `association(aa)` for some `aa` $\in$ `AIdSet`. The same reasoning holds for the number of messages in the communicating state that can be in transit at any moment in time. In order to obtain some of the effects of the message reordering that has to be allowed by the OM/RR protocol, only a few messages need to be considered. Since a distinction has been made between user confirmed and unconfirmed service elements, which represent classes of service elements with equal numbers and types of service primitives, it is only fair to consider at least one message of each of these two

classes to represent *all* messages in transit. As far as the first two problem areas in the specification are concerned a reasonable solution exists. The third point of attention, i.e. the infinitely many sessions an association can have, is a more profound problem. The OM-service layer requires that sessions cannot be distinguished from one-another, and it is up to the OM-protocol entities and the RR-service layer to take care that this is actually the case. Yet, within an OM-protocol entity, a distinction *has* to be made towards different sessions, in order to eliminate the influence past sessions might have on a current session. Here, no real solution can be provided. Although considering only one or two sessions does not satisfactorily abstract from the infinite number of sessions, due to complexity reasons, no other options exist.

Concrete choices that have been made in writing down the dynamic behaviour of the models are listed below.

- Both the OM-service and the OM-protocol entities are restricted to one association only.
- Both the OM-service and the OM-protocol entities support at most three communication messages that can be in transit during the communicating state (see Fig. 5) at any point in time.
- The OM-protocol entities are restricted to *at most* two sessions.
- For the RR-service, the number of messages that can be in transit is reduced to five.

Further simplifications that have been made concerned the data types, written as ACT-ONE abstract data types in LOTOS. In general, the number of service elements was reduced by considering only the different *classes* of service elements instead of every single service element. The tool CÆSAR.ADT was used to compile the abstract data types. In compiling the data types, an anomaly in this tool was discovered. Instead of being able to enumerate the following (enumerable) sort `OSP`:

```
...
SORTS OSP
OPNS enablereq  (*! constructor *) : AID, ID -> OSP
     enableind  (*! constructor *) : AID, ID -> OSP
     enableconf (*! constructor *) : AID, ID, BOOL -> OSP
...
```

CÆSAR.ADT reports a warning about a theoretical limitation stating that no enumeration of this type definition can be made. This is against the intuition since in our case the sorts `AID`, `ID` and `BOOL` are all enumerable and finite. Indeed, an enumeration of the sort `OSP` does exist!

3.4 Verification and Analysis of the Models

The models that were obtained by applying the restrictions in Section 3.3 were analysed with tools from the tool-set EUCALYPTUS, in particular, with CÆSAR, CÆSAR.ADT and ALDÉBARAN. CÆSAR.ADT compiles the data types into a processable form, CÆSAR generates a *labelled transition system* (LTS) from a LOTOS model, which can then be analysed with ALDÉBARAN. The goal was in this case two-fold:

- show absence of deadlock in both the OM-service and in the decomposition consisting of the OM-protocol entities and the RR-service;
- find a formal relation (if one exists!) between the OM-service and its decomposition consisting of the OM-protocol entities and the RR-service, that is as strong as possible, i.e. verify equation (2.1).

The model for the OM-service does not distinguish between different sessions, whereas the model for the OM-protocol does. Essentially, the model for the OM-service describes an unbounded number of sessions and the model for the OM-protocol describes a bounded number of sessions. Hence, only a preorder relation between the two models can be expected.

The first activity was to validate the models against the original LOTOS specifications, using the simulation tool XSIMULATOR, which is comparable to SMILE. Alongside, each time changes were

made to one of the models, absence of deadlock was checked. The final versions of the OM-service model and the OM-protocol model – with, due to complexity reasons, the additional restriction of considering only *one* session – could be proved to have absence of deadlock using ALDÉBARAN. At that point, simulations of the OM-service and the OM-protocol did no longer reveal any strange behaviour. The restriction of the RR-service model to model five messages in transfer proved to be minimal: allowing only four messages in transfer introduced a deadlock in the OM-protocol.

The second objective was achieved by showing that a *safety preorder* [9] exists between the OM-service and the OM-protocol decomposition (again, due to complexity reasons, restricted to only *one* session). This was proved using the tool ALDÉBARAN. This also is the strongest relation that can be shown to exist with ALDÉBARAN between the OM-service and the OM-protocol, as ALDÉBARAN produced counter-examples for stronger relations. Basically, the safety preorder says that the protocol shows only behaviour that is allowed by the service.

A few comments and statistics for the generation of the LTSs are in order. Basically, two methods for generating an LTS from a LOTOS-source are possible: the compositional and the non-compositional method. The non-compositional method is easiest to use, since it requires no insight in the LOTOS-source. However, the drawback may be that the LTSs generated this way are too big to be calculated on most machines. The compositional method on the other hand, allows one to generate LTSs of orthogonal parts of the LOTOS-source and combine these LTSs (or the LTSs that have been reduced to their strong bisimulation normal form) in a later stage. This yields smaller LTSs in usually less time. Decomposing the LOTOS-source into these orthogonal parts, however, can be very difficult and may not always be the most efficient way. In generating LTSs for our models, we used both strategies. The results (obtained on a Linux based Intel Pentium II-300 Mhz, 64 Mb with 128 Mb swap) are listed in Table 2.

	# States	# Transitions	# τ-Transitions	Reduction
OM-service	420,611	2,404,442	n.a.	none
non-compositional	n.a.	n.a.	n.a.	strong
OM-service	97,528	492,714	150,948	none
compositional	52,538	283, 512	80,720	strong
OM-protocol$_1$	n.a.	n.a.	n.a.	none
non-compositional	n.a.	n.a.	n.a.	strong
OM-protocol$_1$	47,525	209,190	25,238	none
compositional	5,180	21,272	14,942	strong
OM-protocol$_2$	n.a.	n.a.	n.a.	none
non-compositional	n.a.	n.a.	n.a.	strong
OM-protocol$_2$	n.a.	n.a.	n.a.	none
compositional	n.a.	n.a.	n.a.	strong

Table 2. *Statistics for the generation of the LTSs for the* LOTOS*-sources. The subscripts* 1 *and* 2 *indicate the number of considered sessions.*

In Table 2, n.a. stands for *not available*. This means that the results of these operations could not be computed due to the limitations on either the hardware, or the available time to do the calculation. For instance, the time spent on calculating the LTS for OM-protocol$_2$ – the LTS for the combination of RR-service and two OM-protocol entities with two sessions – using a compositional method, was aborted after running for more than five days, since CÆSAR reported that still only approximately 65% of the total state space was generated and this figure had been stable for four days. An attempt was made to calculate the LTS for OM-protocol$_2$ on a more powerful machine, however, no significant improvement over the Linux machine was noted.

To illustrate the gain in time that can be achieved using the compositional method over the non-compositional method, consider that the results for the OM-service in a non-compositional

strategy took more than three hours, whereas the compositional strategy took less than 30 minutes on the same machine.

3.5 Specifications and Models

During formalisation, verification and analysis of the OM/RR protocol we made a distinction between a formal *specification* and a formal *model*. They turn out to be related, but different uses of formal methods, which are both useful.

Specifications are meant to prescribe the behaviour of a system. They serve as the basis for implementation, coding and testing. Consequently, specifications should be as complete as possible, taking into account all possibilities of behaviour. On the other hand, models are meant to focus on one particular aspect of a system while abstracting from other aspects.

Specifications also abstract from many details, but these details concern properties which are not considered at all. For a given level of abstraction, or for a given family of properties under consideration, specifications should be complete. Models, however, also abstract from properties which are, in principle, under consideration. Consider, for instance, real-time properties of the OM/RR protocol: they are not considered at all in our analysis and they appear neither in our specifications nor in our models. On the other hand, in principle, we are interested in the behaviour of protocol entities in the presence of 5 other protocol entities and communicating via 9 associations. This behaviour is described in the formal specification, but for the models we made simplifications and restricted them to one other protocol entity communicating via only one association.

A specification is often too complex and contains too much detail for checking of properties, e.g. model checking. This is especially true if property checking is performed using a tool; state of the art tools, like EUCALYPTUS, cannot deal with large and complex formal descriptions; restrictions and simplifications are necessary. Hence, for checking of properties a formal model is developed. A model is a simplification of the system in which a lot of detail has been removed and only those aspects which are deemed important for the property at hand, are formally expressed. But, because of this, a model cannot be used, e.g. as the basis for testing of implementations: since some allowed behaviours may have been removed for simplification it cannot be decided for actions executed by an implementation whether they are allowed or not.

Making models is an intricate process in which the right level of abstraction must be chosen to achieve a good compromise between simplicity and completeness. As a consequence, verifying a property for a model does not give certainty about the system: any verification is only as good as the validity of the model on which it is based. Validity of the model is usually assumed or informally, using hand waving arguments, reasoned upon.

For the OM/RR protocol we first developed complete specifications (with respect to the properties under consideration), see Section 2.3. These specifications were analysed using tools from LITE. These tools have low functionality but do not impose restrictions on the formal specifications. Based on the specifications, simplifications were made such as restricting the number of associations, sessions, protocol entities, different kinds of messages, etc., see Section 3.3. This led to models which were amenable for the EUCALYPTUS tools. Properties were checked on these models but no certainty is obtained that any checked property, e.g. deadlock freedom, also holds for the complete specification. Only informal and hand waving arguments were given in Section 3.3.

When constructing specifications and models, different approaches are possible. On the one hand, one can start by first constructing the complete specification and then try to validate this specification. This validation may involve constructing models from the specification and verify these against desired properties. Another possibility is to consider models of the vital behaviour first and prove them correct by means of verification, and gradually extending these models to become a specification. Finally, a mixture of both approaches can be used, by constructing models and a specification hand in hand. The first approach is the one used in this project. Although very suitable, it is felt that the third approach would have been more convenient in the end.

4 Conclusions

The results described in this paper deal with a case-study of an industrial *mission critical* system, called the *Operator Support System* (OSS). Various inherently complex communication protocols, part of the OSS, are potential hazards for the correct functioning of the OSS. Because of the mission critical aspects of the OSS, CMG Den Haag B.V. decided to include formal methods in the development of the OSS. This paper describes the formalisation, analysis and verification of one of these protocols, the OM/RR protocol, using the *Formal Description Technique* (FDT) LOTOS. The number of hours spent on this project roughly equals 9 man months. This time includes the learning time of the FDT LOTOS and familiarising with the OSS project and the tool-sets.

As we showed, the trajectory of formalising an informal document itself already reveals many omissions, ambiguities and errors; as such, formalising informal documents is beneficial, even without doing anything with the resulting formal specification. Due to the ambiguities and omissions, the real functionality of the OM/RR protocol is shrouded in mist. This makes it hard to judge the quality of the protocol itself. Since the OM/RR protocol serves as one of the backbones of the OSS, it forms a severe risk for the reliability of the OSS.

Despite the severe omissions and ambiguities in the informal document we tried to develop a formal specification of the OM/RR protocol. Some of the ambiguities could be eliminated by taking the design requirements into account. However, the general lack of documentation concerning the design requirements did not allow for such solutions in general. Instead, conversations with the developers of the protocol served to find out about the assumptions and design requirements that were made during the trajectory of writing the formal specifications. These conversations revealed that the developers considered the dynamic behaviour to be trivial. As such, it had not been included in official documents. Formalising the behaviour of the protocol, however, raised many questions. This implies the dynamic behaviour cannot be considered trivial.

The conversations with the developers can be characterised as conversations not so much about design and specification, as about implementation. Our level of abstraction of talking about the protocol differed from the level of the developers. Our interest mainly focussed on the *service* of the protocol, yet, the developers mainly focussed on how to overcome various problems when *implementing* the protocol in real-life. Retrieving information and getting to the essence of the matter proved harder than was expected.

Although some of the questions that were found in the trajectory of formalising the OM/RR protocol have been answered, many questions remained open. As a result, the formal specification of the OM/RR protocol can only be considered as a first attempt. There is no guarantee at all that our formal description corresponds with the intentions of the developers. Consequently, the greatest benefit that is obtained in this project is not in the formal specifications, but in the number of questions that have been raised in the process of formalising, and that can be a basis for a next version of the protocol. With respect to the OM/RR protocol, the conclusion of the formalisation project is that the current document as it is [7], is not a good basis for implementing the crucial OM/RR protocol; it must certainly be improved.

The FDT LOTOS turned out not to be of great help in clarifying various questions posed to the developers of [7]. The reason was that the LOTOS descriptions were not really understood by them. An explanation for this can be found in the style that was used to specify the system, i.e. a *constraint-oriented* specification style [22], and their inexperience with the FDT LOTOS. This style is very useful in specifying a system, however, reading a specification in this style puts an extra burden on the reader, since it requires a reader to combine various constraints found throughout the specification.

Another issue can be that a process algebraic approach might not be the most suited for use in an industrial environment when confronted with people with no clear background in formal methods. Case studies are required, in which various formal methods are used to tackle a problem, in order to determine the most suited method (both in apprehension and application) for various problems. It is expected that visual formal methods, e.g. SDL [17] or MSC [18], might be more suited when explaining ambiguities and omissions to people with less experience in formal methods.

It can also be very beneficial to present the results of validations in a visual formalism. For instance, the traces of LOTOS simulations could be presented as MSC. This was used successfully in the BOS project (cf. SPIN [15] and [10]). Complete specifications in visually oriented languages, however, are thought to be hampered by the same problem of complexity. An interesting exercise would be to investigate the time it takes for people to read and fully understand various specifications in various formalisms. While LOTOS was not easily understandable by the developers of the protocol, the people doing the formalisation were reasonably satisfied. There are some anomalies in the language, e.g. see Section 3.1, but as a whole LOTOS is useful.

Although a full-fledged specification is inevitable when designing protocols, a fully *formal* specification of dynamic behaviour might not be most useful. On the one hand, writing a very detailed formal specification is a very cumbersome task, taking a lot of time, and on the other hand, vital parts of a specification are less likely to be recognised within a very detailed specification. As such, formal models or partial formal specifications may be more suited.

The tool-sets used in our project, EUCALYPTUS and LITE, are useful for this kind of projects. Confidence in a specification is only achieved by validation and verification. For any realistically sized formal description this cannot be done manually; tools are indispensable. The simulator SMILE contained in LITE turned out to be useful, mainly in the initial phases of development, for analysing specifications, but its usage was rather cumbersome and laborious. CÆSAR and ALDÉBARAN contained in EUCALYPTUS were mainly used for checking properties on models. EUCALYPTUS was found to have an omission concerning the abstract data types in the tool CÆSAR.ADT, see Section 3.3. Moreover, a thorough investigation concerning the efficiency of this tool-set with respect to memory usage is needed. Memory overflow, caused by state-space explosion, is one of the best known and most studied problems restricting industrial usage of this kind of tools. It would be interesting to investigate the use of the compositional approach in order to be able to give some general guidelines for optimal usage of this approach, and ultimately automate this approach. Last but not least, feedback from tools is an important issue, e.g. a deadlock found in a labelled transition system, is hard to locate in the corresponding LOTOS model. A tool helping to bridge the gap between these two representations would have saved much time.

Acknowledgements

The first author would like to thank Jos Baeten of the Eindhoven University of Technology for the supervision during this project. We thank the members of the SIG-MCS/FM group of CMG, especially Michel Chaudron and Bart Botma for the numerous discussions on any topic and Jan Friso Groote for proof-reading early versions of this paper. CMG Den Haag B.V. is thanked for the pleasant working atmosphere and the financial support during this project.

References

1. ISO/IEC 10164. *Systems Management*. Geneva, 1990.
2. ISO/IEC 8807. *Information Processing Systems, Open Systems Interconnection, LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*. Geneva, 1989.
3. ISO/IEC 8824. *Information Processing Systems - Open Systems Interconnection, Specification of Abstract Syntax Notation One (ASN.1)*. Geneva, March 1988.
4. ISO/IEC 9072. *Remote Operations*. Geneva, 1990.
5. ISO/IEC 9595. *Common Management Information Service Definition*. Geneva, 1991.
6. ISO/IEC 9596. *Common Management Information Protocol Specification*. Geneva, 1991.
7. Adviesdienst voor Verkeer en Vervoer. *OM/RR and Association Management Specifications, version 1.2*, July 18th 1997.
8. T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. In *Computer Networks and ISDN Systems*, volume 14, pages 25–59, 1987.
9. A. Bouajjani, J. C. Fernandez, S. Graf, C. Rodriguez, and J. Sifakis. Safety for Branching Time Semantics. In *18th ICALP*. Springer-Verlag, 1991.

10. M. Chaudron, J. Tretmans, and K. Wijbrans. Lessons from the Application of Formal Methods to the Design of a Storm Surge Barrier Control System. In J.M. Wing, J. Woodcock, and J. Davies, editors, *FM'99 – World Congress on Formal Methods in the Development of Computing Systems II*, volume 1709 of *Lecture Notes in Computer Science*, pages 1511–1526. Springer-Verlag, 1999.
11. R. De Nicola. Extensional equivalences for transition systems. *Acta Informatica*, 24:211–237, 1987.
12. H. Eertink. Executing LOTOS specifications: The SMILE tool. In T. Bolognesi, J. van de Lagemaat, and C. Vissers, editors, *LOTOSphere: Software Development with LOTOS*, pages 221–234. Kluwer Academic Publishers, 1995.
13. H. Garavel. OPEN/CAESAR: An Open Software Architecture for Verification, Simulation, and Testing. In B. Steffen, editor, *Proceedings of the Fourth International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'98*, volume 1384 of *Lecture Notes in Computer Science*, pages 68–84. Springer-Verlag, March 1998.
14. Hubert Garavel. An overview of the EUCALYPTUS toolbox. In Z. Brezočnik and T. Kapus, editors, *Proceedings of the COST 247 International Workshop on Applied Formal Methods in System Design (Maribor, Slovenia)*, pages 76–88. University of Maribor, 1996.
15. G.J. Holzmann. The Model Checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.
16. ISO. *Information Processing Systems, Open Systems Interconnection, Basic Reference Model.* International Standard IS-7498. ISO, Geneve, 1984.
17. ITU-T. *ITU-T Recommendation Z.100: Specification and Description Language (SDL).* ITU-T, Geneva, 1992.
18. ITU-T. *ITU-T Recommendation Z.120: Message Sequence Chart (MSC).* ITU-T, Geneva, 1996.
19. D. Steedman. *Abstract Syntax Notation One (ASN.1) - The Tutorial and Reference.* Isleworth: Technology Appraisals, 1990.
20. J. Tretmans, K. Wijbrans, and M. Chaudron. Software Engineering with Formal Methods: The Development of a Storm Surge Barrier Control System – Seven Myths of Formal Methods Revisited. In S. Gnesi and D. Latella, editors, *Fourth Int. ERCIM Workshop on Formal Methods for Industrial Critical Systems (FMICS'99) – Proceedings of the FLoC Workshop*, volume II, pages 225–237, Pisa, Italy, July 11–12 1999. Servizio Tecnografico Area di Ricerca del CNR.
21. C. A. Visser and L. Logrippo. The Importance of the Service Concept in the Design of Data Communications Protocols. In M. Diaz, editor, *Protocol Specification, Testing, and Verification V*, pages 3–17. Elsevier Science Publishers B.V. (North-Holland), 1986.
22. C.A. Vissers, G. Scollo, M. van Sinderen, and E. Brinksma. Specification styles in distributed systems design and verification. *Theoretical Computer Science*, 89:179–206, 1991.
23. T.A.C. Willemse. The Specification and Validation of the OM/RR-Protocol. Master's thesis, Eindhoven University of Technology, July 1998.