A Formalization of a Home Banking Protocol Using VDM

Simon Zwölfer[1] and Brigitte Fröhlich[2]

[1] Technical University Graz, Austria,
szwoel@iicm.edu
[2] Siemens AG Austria, Graz
brigitte.froehlich@siemens.at

1 Introduction

During the last years communication protocols played an important role in our daily work with the computer. For our personal use, especially when we use applications for electronic banking, we assume that such protocols are at least reliable and secure. For distributed systems in general many other properties are desired. But how can developers assure these properties?

Different organizations like the CCITT or the ISO are making efforts in the development of so-called *Formal Description Techniques (FDTs)* such as LOTOS, ESTELLE and SDL. On the one hand these techniques can play a part in the system's development process and on the other hand they offer concise, unambiguous and implementation independent descriptions of protocols and concurrent systems. These descriptions are then subject to a rigorous analysis and verification step, which can be done in various levels of rigor and formality [4].

The German Central Finance Committee (ZKA) in Germany introduced a protocol called *Home-Banking-Computer-Interface (HBCI)* in a minor rigorous way — in form of an informal specification [2]. Since the first release in 1996 this document has grown in both volume and complexity and it seems that much effort was necessary to keep a specification of about 550 pages up to date. But what problems arose during this work? What are the reader's problems with such specifications? Would it be better to define parts of the specification more formally?

Some of these questions were answered during the work on a diploma thesis initiated by the Institute of Software Technology, Technical University Graz (IST), and the Siemens Program and System Development, Graz (PSE). At the time of Version 1 of the HBCI standard the people at Siemens tried to read and understand the informal specification with less effort. But too many details seemed to overwhelm the readers when they tried to find answers for basic questions. So one tried to formalize some of the syntactical definitions by using a BNF. But it did not work — why? To gain an answer to this and many other questions the project was initiated.

2 Why VDM? A Language Oriented Approach

One of the first decisions to be made before a problem can be formally described is the choice of the development method and its specification language. Many factors like system requirements, demands concerning verification and validation, and the structure or class of the problem may assist in this decision.

The Vienna Development Method (VDM) and its specification language (VDM-SL) is not a typical method to describe communication protocols in general. A vast variety of specification languages has been designed for this purpose [1]. But what makes VDM a possible candidate for the specification of the HBCI protocol?

First of all, the structure of the informal HBCI specification invites to try a language oriented approach. Tables containing syntactic definitions are used in the whole document. Each row of such a table contains a new syntactical construct (e.g. for basic types like numeric types, which are comparable to types in programming languages) or it refers to a predefined syntactical construct. These definitions form the basis for informal descriptions, which are given for both single rows in the tables and the table itself. No matter which approach or method is actually chosen it is obviously necessary to provide the syntactical definitions for the construction of *messages* which are actually delivered between the communication parties.

Another argument can be taken from [3] or [5]. Here the *relation* between protocols and languages is emphasized. A protocol can be defined by means of five distinct parts. The *service* to be provided by the protocol (i.e. transfer of files, bank transactions) and assumptions about the *environment* (i.e. the communication channel) define the environment where the protocol is used. Further the *vocabulary* of messages (semantics), their *encoding* or format of a message in the vocabulary (syntax), and finally the *procedure rules* describe how information can be encoded and transmitted (grammar). The last three items — vocabulary, encoding, and procedure rules — can be part of a language definition. For each receiving party the protocol defines: "a language, whose sentences are the legal sequences of messages received by that party, and whose alphabet of symbols is the set of all possible messages".

In general one part of a language specification describes semantics. The semantics of protocols can be described in different ways. In CSP for example, the external behavior of such a system is described. These models are then verified for the presence of special conditions like deadlock or livelock. In our case the protocol flow is rather simple and these properties are not in the fore. The semantics might be restricted to the correctness of given messages in their context which is also a question in a conventional specification of programming languages.

Hence, these techniques were preferred in the specification of the HBCI protocol and VDM-SL, a widely used method for this class of problems, was applied.

3 Structure of the Specification

The specification of languages as introduced in the last section is first of all applied to messages of the HBCI protocol. We will see that this approach is suitable for the description of the basic structures. Semantics, which is often described in two different steps, is not described exactly as this is done in programming languages. As mentioned above not the meaning (or consequences) of messages and the embedded transactions for an individual customer's account is the major point of interest in the specification of the HBCI protocol. On the countrary, the specification is focused on the static semantics of a sequence of messages.

First of all, a short overview of the syntax and its description is given.

3.1 Syntax

As described above one part of a language specification is the declaration of the syntax — in the case of HBCI the syntax of messages and their subordinate units.

In the HBCI protocol the syntax is derived from the UN/Edifact standard. A rigorous analysis of the syntax includes a proof of unambiguousness. In a first step the detailed syntax was defined by means of BNF grammar rules. These rules can be used to analyze or synthesize messages. All necessary parts beginning with basic types and ending with whole messages are declared here.

Syntax declarations serve two purposes in the formal HBCI specification. On the one hand they show how HBCI messages are constructed. On the other hand they can be used in a so-called *Syntax Directed Translation* to define how the message can be translated into a unique VDM representation. One drawback of BNF description is the boundary where a context must be taken into consideration. Contextual conditions are not declared in the grammar but within *contextual constraints*. The first class of contextual constraints appears within the syntax directed translation rules in form of VDM data type *invariants*.

3.2 Contextual Constraints

In general semantics of programming languages assigns a meaning to grammatically correct programs. But the semantical correctness of a program is not sufficient for having a meaning. Additional constraints as for example typing or scoping rules must be met by the program. Usually a well-formedness function is related to each phrase ensuring that the static conditions are met. But how can such an approach be used for the HBCI protocol?

Beginning with an abstract syntax in terms of VDM types a basic concept for HBCI messages exists. The informal specification defines a lot of rules for the validness or *well-formedness* of messages and subordinate syntactical elements. Applying the approach above we have to classify the rules into different categories.

Type Rules. Which have been defined in the syntax declarations.

Message Rules. These rules are contextual constraints, which are applied within a message. They define the conditions under which syntactic elements are correct where only the other syntactic elements of the message are taken into consideration. For example a message contains a header and a trailer. Both header and trailer have an identifier, which must be equal. This condition cannot be defined by means of a BNF rule.

Dialog Rules. At this level of correctness the connection between message parts is considered. Here for example is the semantics of special error-codes defined.

Bank Semantics. Obviously each message contains some transactions which are expected to have some consequences to bank accounts etc. But this kind of semantics is neither defined exactly in the formal nor in the informal specification.

3.3 Protocol Semantics

The specifications of the syntax and the contextual constraints define the framework where these transactions and queries can be performed — the *protocol semantics* which concerns only the levels mentioned above. Hence, protocol semantics describes all mechanisms which are provided by the protocol to guarantee a safe and secure communication. It defines typical elements of a protocol such as counters, checksums, version identifiers, etc. but also elements like the customer-id and the bank-id which have well defined consequences for both the protocol flow and the bank transactions.

The HBCI protocol defines a dialog between a customer and a bank. A HBCI dialog consists of a sequence of *Dialog-Steps*. Each step has exactly two HBCI-Messages. One of these messages is send by the customer and received by the bank through a communication channel. Another channel is used by the bank to send an answer back to the customer. Additionally, the dialog is divided into three phase: the initialization-phase, the transaction-phase and the conclusion-phase.

$$\begin{aligned} \mathit{Dialog} :: & \mathit{initialization\text{-}phase} : \mathit{DialogStep} \\ & \mathit{transaction\text{-}phase} : \mathit{DialogStep}^* \\ & \mathit{conclusion\text{-}phase} : \mathit{DialogStep} \end{aligned}$$

$$\begin{aligned} \mathit{DialogStep} :: & \mathit{customer} : \mathit{Message} \\ & \mathit{bank} : \mathit{Message} \end{aligned}$$

A dialog happens during a physical connection which provides a virtual point-to-point connection residing on the transport layer of the ISO/OSI reference models. The VDM type *PL-Session* models the properties of such a physical connection and its dialog. The *Bank ID* and the *line parameters* define the chosen credit institute and connection type. Messages which denote communication events are exchanged during the physical session.

The VDM type *PL-Message* models both customer and bank messages on the physical level. Each message has three important properties. The *origin*, which denotes the sender of a message, the *content*, which is represented by a series of a characters, and finally the *transmission state*, which tells whether the message content has been actually received by the other party or has been lost in the network.

$$
\begin{array}{l}
PL\text{-}Message :: origin : Origin \\
\qquad\qquad\quad cont : \mathsf{char}^{+} \\
\qquad\qquad\quad tr\text{-}state : PENDING \mid RECEIVED
\end{array}
$$

$$
\begin{array}{l}
PL\text{-}Session :: bank\text{-}id : BankId \\
\qquad\qquad\quad line\text{-}par : LinePar \\
\qquad\qquad\quad msgs : PL\text{-}Message^{*}
\end{array}
$$

$$
\begin{array}{l}
\mathsf{inv}\ \mathsf{mk}\text{-}PL\text{-}Session\ (\text{-}, \text{-}, msgs) \triangleq \\
\quad (\forall\, i, j \in \mathsf{inds}\ msgs \cdot \\
\qquad (i = j\text{-}1 \ \Rightarrow\ msgs\,(i).origin \neq msgs\,(j).origin) \wedge \\
\qquad (i = 1 \ \Rightarrow\ msgs\,(i).origin = CUST)) \wedge \\
\quad (\forall\, i, j \in \mathsf{inds}\ msgs \cdot \\
\qquad (i < \mathsf{len}\ (msgs) \ \Rightarrow\ msgs\,(i).tr\text{-}state = RECEIVED));
\end{array}
$$

Compared to the abstract syntax of a HBCI dialog no explicit subdivision of dialogs in dialog steps can be found in the type *PL-Session*. Dialog steps in general are modeled by means of the invariant. The explicit subdivision of a dialog in its three phases is the modeled by corresponding VDM state variables.

$$
\begin{array}{l}
wf\text{-}PL\text{-}Sessions\,(r : PL\text{-}Session)\ r : \mathbb{B} \triangleq \\
\quad (LCTX\text{-}.bank\text{-}id := s.bank\text{-}id; \\
\quad\ LCTX\text{-}.line\text{-}par := s.line\text{-}par; \\
\quad\ DCTX\text{-} := INIT\text{-}DCTX; \\
\quad\ \mathsf{for}\ cur\text{-}msg\ \mathsf{in}\ s.msgs \\
\quad\ \mathsf{do}\ (\mathsf{if}\ (cur\text{-}msg.origin = CUST) \\
\qquad\ \mathsf{then}\ \mathsf{if}\ (\neg\ wf\text{-}PL\text{-}CMsg\ (cur\text{-}msg)) \\
\qquad\qquad\ \mathsf{then}\ \mathsf{return}\ \mathsf{false} \\
\qquad\qquad\ \mathsf{else}\ \mathsf{skip} \\
\qquad\ \mathsf{else}\ \mathsf{if}\ (\neg\ wf\text{-}PL\text{-}MBsg\ (cur\text{-}msg)) \\
\qquad\qquad\ \mathsf{then}\ \mathsf{return}\ \mathsf{false} \\
\qquad\qquad\ \mathsf{else}\ \mathsf{skip})) \\
\mathsf{ext}\ \mathsf{wr}\ LCTX\text{-} : LCtx \\
\quad\ \mathsf{wr}\ DCTX\text{-} : DCtx
\end{array}
$$

The operation *wf-PL-Sessions* is used to verify the correctness of sessions and their dialogs. It consists of two parts. In the initialization part the dialog parameters are assigned to corresponding values in the *line context* part of the global state and the *dialog context* is initialized. The second part checks the correctness at the level of messages (i.e. the message is correct with respect to

its source). All parameters which influence a HBCI dialog are settled in the VDM state of the dialog description. Apart from the line context and the dialog context the parameters include general protocol properties, a customer environment, a bank environment, and a history of dialog contexts.

4 Results of the Specification

The result of a formal specification is obviously not the specification alone. There are many "by-products" which can be of more or less importance [6]. One of these by-products of the formal HBCI specification is a list of problems and questions with the informal one. This list contains many aspects where the informal document seems to be not exact enough or too much freedom is left to the developer.

But what happened to the goals which have been defined at the beginning of the work? Two of them should be mentioned in this scope. An important question of Siemens was whether or not the informal document contains ambiguities, mistakes or if any point has been left out. These questions were mostly answered by means of the list mentioned above.

Another question concerned the freedom of the developers and institutes to define institute specific transaction types, which exist beside the HBCI standard without any standardization. This question has its origin in the history of protocols defined by the ZKA. On the one hand such freedom is necessary to temporally enlarge the functionality of the protocol but on the other hand an exhaustive use of this possibility would lead to "institute specific standards". The specification gives an affirmative answer to this question.

Last but not least the formal specification may serve as a rigor definition of the basic protocol functionality. It may exist beside the informal document, which gives information about the bank semantical background and bank semantical conditions which are beyond the scope of the formal document.

References

1. Daniel Cooke, Ann Gates, Elif Demirörs, Onur Demirörs, Murat M. Tanik, and Bernd Krämer. Languages for the specification of software. *Journal of Systems and Software*, 32:269–308, 1996. ISSN:0164-1212.
2. HBCI – homebanking computer interface: Schnittstellenspezifikation 2.0.1, February 1998. Bundesverband Deutscher Banken, http://www.siz.de/siz/hbci/index.html.
3. Gerhard J. Holzmann. *Design and Validation of Computer Protocols.* Series in Computer Science. Prentice Hall, Englewood Cliffs, N.J., 1991. http://cm.bell-labs.com/cm/cs/who/gerard/popd.html.
4. John Rushby. Formal methods and their role in the certification of critical systems. Technical Report SRI-CSL-95-1, Computer Science Laboratory, SRI International, Menlo Park, CA, March 1995.
5. Robin Sharp. *Principles of Protocol Design.* Series in Computer Science. Prentice Hall, United Kingdom, 1994. ISBN: 0-13-182155-5.
6. Jeanette M. Wing. A specifier's introduction to formal methods. *IEEE Computer*, 23(9):8, 10–22, 24, September 1990. ISSN: 0018-9162.