Application of a Formal Specification Language in the Development of the “Mobile FeliCa” IC Chip Firmware for Embedding in Mobile Phone

Taro.Kurita@jp.sony.com Yasumasa.Nakatsugawa Miki.Chiba

Abstract

We applied the formal specification language VDM++ in the development of firmware of the “Mobile FeliCa” IC chip. As an outcome, we have achieved successful results and confirmed its effectiveness.

The objectives of applying a formal method were as follows: (1) Description of rigorous specifications; (2) Development and application of a scheme and processes for specification development, firmware implementation and testing; (3) Enhancing the quality of deliverables at the upper stream of development process; (4) Testing thoroughly with formal specifications for whole software development processes; (5) Improvement of communication between engineers.

1 Outline of Project

“FeliCa” is a contactless IC card technology widely used in Japan, developed and promoted by Sony Corporation. This FeliCa technology is utilized in the Mobile FeliCa IC chip which is embedded in a mobile phone.

Mobile phones embedded with the FeliCa IC chip are known as “Osaifu Keitai” (means of mobile wallet) by NTT DoCoMo, Inc., and today those chips are embedded in over 50 million mobile phones which can be used as electric money, train tickets, identifications, door keys and so on.

The mobile FeliCa system is comprised of mobile phones with the FeliCa IC chip, FeliCa servers connected to the mobile telecom network and FeliCa reader/writers.

The mobile FeliCa system is shown in Figure 1.

The characteristics of the Mobile FeliCa IC chip firmware are as follows:

- Contains the secure file system and the communications protocol, which are the basis of the FeliCa technology;

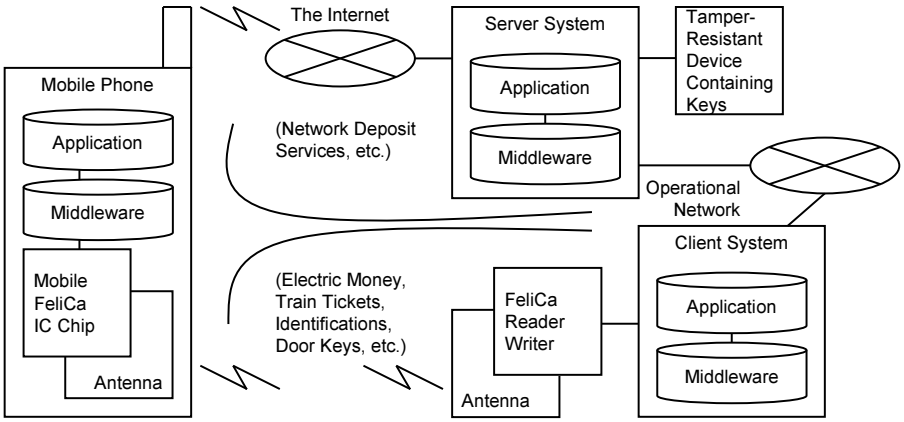

Fig. 1 Mobile FeliCa System

- Possesses firewall functions that enable the multiple services in the Mobile FeliCa IC chip such as electric money and train tickets.

We must ensure the extremely high quality of the software in order to avoid serious problems related to social infrastructure, and so the stakeholders will not be affected. It would be impossible to recall all the FeliCa IC chips which are embedded in mobile phones.

The project duration was three years and three months. There were 50-60 members within this project, and the average age was about 30 years old. There were no members who had the knowledge of or the experience with formal methods at the time of project launch.

We have employed several chip manufacturers in order to reduce risks in manufacturing and sales. It was necessary that the firmware on the different ASICs and firmware development environments behave exactly the same so that the compatibility was retained.

C/C++ and assembler languages are used in the implementation of the Mobile FeliCa IC chip firmware.

2 Objectives

After a consideration to the characterization of the development of the Mobile FeliCa IC chip, we decided to focus on improvements at the upper stream of development process related to software development and mutual understanding between engineers, and we have taken on the challenge of describing formal specifications.

The objectives of applying a formal method were as follows:

1. Description of rigorous specifications and defining functions;
2. Development and application of a scheme and processes for specification development, firmware implementation and testing;
3. Enhancing the quality of deliverables at the upper stream of development process;
4. Testing thoroughly with formal specifications for whole software development processes;
5. Improvement of communications between engineers.

3 Approach

We have decided to use the formal specification language VDM++[1] and VDMTools [3], since they support describing and executing large-scale models.

The obtained results of the overall development process are shown in Figure 2. We have developed and tested external specifications using VDM++.

The process of specification development is as follows:

1. Discussed requirements with stakeholders and wrote the general specifications in a natural language with various diagrams based on UML notation, such as state transition diagrams and sequence diagrams;
2. Modeled the FeliCa file system and designed and implemented a framework in VDM++;
3. Described command and security specifications using the framework;
4. Tested specifications using a unit testing framework.

The main components or functions of the formal specifications are as follows:

- The FeliCa file system specification that defines the basic data structure;
- The framework for describing and testing specifications that are based on the basic data structure;
- Command specifications which are the basis of the FeliCa technology;
- Security specifications.

Non-functional specifications such as performance and reliability were written in the natural language separately from the formal specifications.

VDM++ is a multi-purpose formal specification language that is primarily an object-oriented extension of VDM-SL [2] which is a formal specification language standardized under the International Organization for Standardization (ISO).

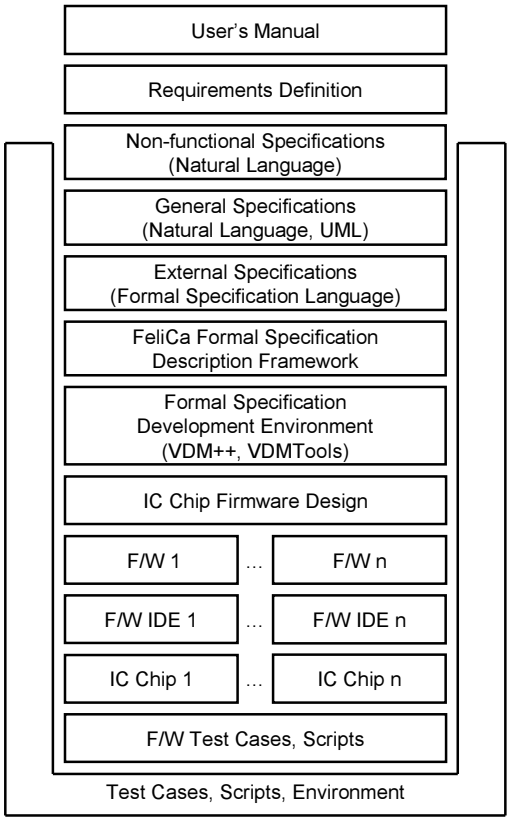

Fig. 2 Obtained Results

VDMTools is an integrated tool that supports model analysis, specification description and testing. We have used following functions; (1) syntax checking, (2) type checking, (3) sequential implementation of executable specification and debugging support, (4) measurement of the code coverage of executable specification.

VDMTools has the conversion function of executable specification to C++ or Java code, however because the generated code was not suitable for the embedded development environment for secure IC cards, this function was not used.

In the project, we organized three teams; specification team, firmware implementation team and testing team. There were 5-20 members, 15-20 members and 25-35 members respectively.

The outline of the development teams is shown in Figure 3.

The outline of the overall development process is shown in Figure 4.

Considering a single iteration starts from describing specifications to testing the firmware on development board, one iteration consumes a few weeks, and in total 30 iterations were carried out.

4 Test Scheme

Test scheme for the overall development is shown in Figure 5.

From the formal specifications, test engineers designed black-box test specifications and then implemented test scripts. Executable formal specifications, firmware on development board and IC chips were automatically tested using the test environment and the test scripts.

From the results of the tests, we were able to confirm whether the test cases and scripts were consistent with the specifications. In addition, from measuring the coverage of the executable formal specifications, we have confirmed that the test cases covered all the defined specifications.

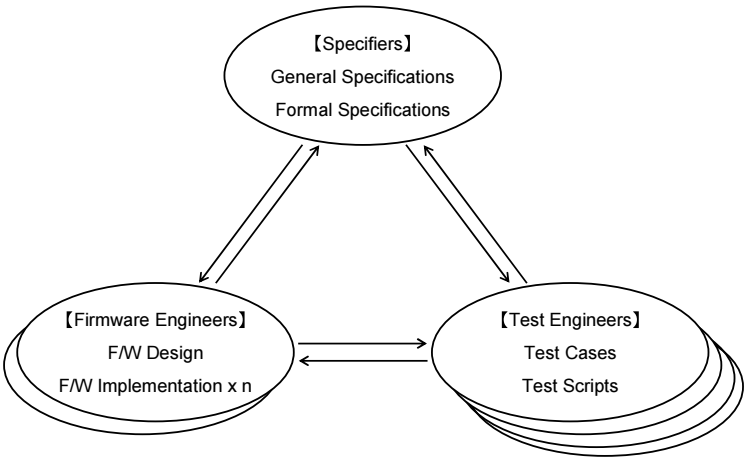

Fig. 3 Development Teams

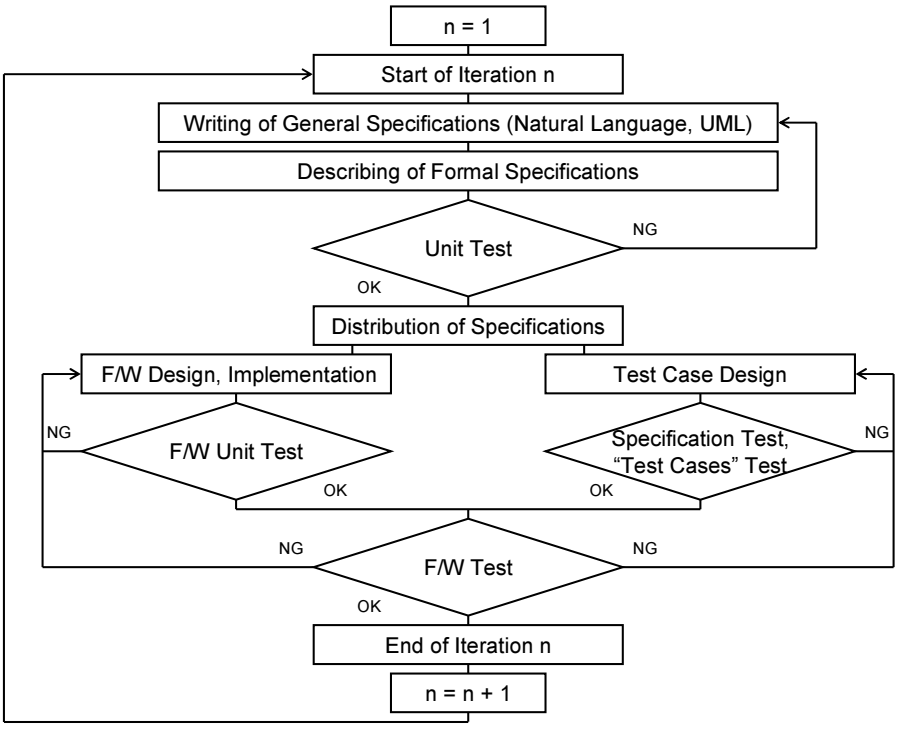

Fig. 4 Development Process

5 Results and Discussion

5.1 Specification Development

At first, we discussed requirements with stakeholders and wrote the general specification in a natural language with various diagrams based on UML notation, such as state transition diagrams and sequence diagrams. Secondly, we modeled the FeliCa file system and, designed and implemented the framework. Finally, we described command

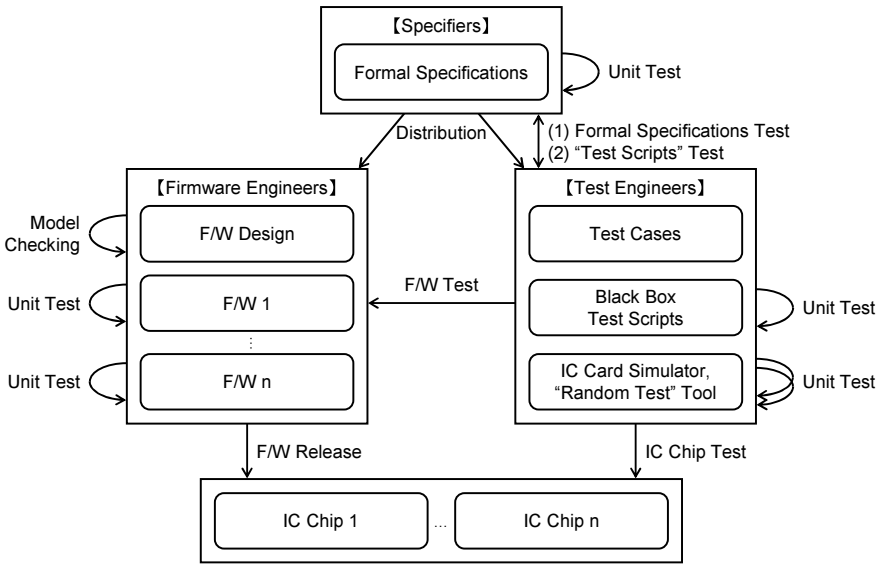

Fig. 5 Test Scheme

Table. 1 Percentages of Errors in Firmware Implementation

Reason for Errors	Percentage
Missing description	0.2%
Erroneous description	0%
Unclear description	1.8%
Oversight	5.6%
Insufficient understanding	10.7%
Insufficient confirmation	0%
Failure of change propagation	0.2%
Others (reasons unrelated to specifications)	81.5%

specifications and security specifications on framework.

Furthermore, we developed class methods for unit testing of formal specification, and tested executable formal specifications.

The results related to specifications are as follows:

- 383 pages of a protocol manual written in the natural language (user's manual for other departments within the company and for outside customers);
- 677 pages of an external specification document written in the formal specification language.

Our formal specifications are about 100,000 steps including test cases (about 60,000 steps) and comments written in the natural language. Using this specifications, we implemented the C/C++ code of about 110,000 steps, inclusive of comments, as firmware of a single IC chip.

5.2 Percentages of Errors in Firmware Implementation

The percentages of errors in firmware implementation related to specifications for the overall project are as shown in Table 1.

The formal methods are useful for finding errors in the early stages of development. The formal specification can be syntax and type checked. In addition, since the specification is executable, testing can be carried out.

From the above results, it can be said that we have successfully described the specifications in a precise way.

Table. 2 Number of Changes

Trigger of Modification	Number	
Additions and modifications by specifiers themselves	84	(46%)
Requests, indications and proposals from firmware engineers	25	(14%)
Requests, indications and proposals from test engineers	23	(13%)
Others	9	(4%)
Requests and proposals from outside the project	41	(23%)
Total	182	

Table. 3 Number of Errors Discovered before Integration Test

Phase of Development Process	Number
Describing Specifications	162
Executing and Unit Testing Specifications	116
Reviewing Specifications	93
Communicating with Firmware Engineers	69
Total	440

On the other hand, the total percentage of “oversight” errors and “insufficient understanding” errors was 16.3%. This was due to the fact that the separations between the actual specifications and the code required to execute the specifications was unclear.

5.3 Efficiency of Specification Development

The average productivity of VDM code for the formal specifications was about 1,900 lines per engineer per month (approximately 160 hours). This number is equal to that for firmware implementation. It can be said that there were no particular disadvantages by using the formal specification language from scratch.

As for the skill level of the engineers, although all has learned the basic information technology, there was a wide gap in the number of years that the engineers had experienced in software development. There were even some members who had no experience.

5.4 Changes of Specifications

182 modifications were made to the formal specifications. Specifiers added, modified and debugged 84 times since formal specification version 1.0 was released. The number of changes are as shown in Table 2.

5.5 Tests and Reviews of Specifications

The line coverage rate of the formal specifications by unit testing was 82%. We were able to enhance the coverage rate of unit test cases by coverage analysis. As a result of unit testing, we were able, for example, to discover an incorrect path in postconditions. This kind of inconsistency is generally difficult to discover by review.

Discovered formal specification errors before integration tests are as shown in Table 3. The formal method contributes to enhancing the quality of deliverables at the upper stream of development process.

The line coverage rate of the formal specifications by black-box testing and visual inspection was 100%.

“Random Test” is an aging test. The test tool sends random commands continuously to the test target and checks whether the test target sends back correct responses.

By carrying out about 7,000 black-box tests and 100 million random tests, the high quality of IC chips was achieved.

Table. 4 Results of Questionnaire

Responses	Specifiers	Firmware Engineers	Test Engineers	Overall
I frequently refer to it	87%	13%	27%	24%
I refer to it	13%	27%	46%	35%
It is easy to read	29%	13%	12%	12%
It is not easy to read, but I can read	71%	53%	67%	59%
It was very good that we applied it	43%	20%	8%	14%
It was good that we applied it	57%	7%	69%	43%
There was no effect observed	0%	14%	4%	6%
A specification language is necessary	87%	40%	81%	65%
I would like to utilize it in the future as well	0%	13%	19%	18%
I would like to utilize it depending on the case	100%	40%	69%	59%

5.6 Formal Specifications and Communication

We have analyzed all the questions related to the specifications from firmware engineers, test engineers and stakeholders. We have divided questions into three categories: “Comprehension”, “Intent” and “Error”.

As compared with the formal specifications, there are more requests for clarification on the general specifications and the manuals written in the natural language. On the other hand, there are more questions related to background comprehension of the formal specifications.

This result shows that the specifications in the natural language were not precise. For the formal language, the backgrounds of the specifications were unclear to readers. Therefore, it is preferable that the background of the specifications written in the natural language are included as comments in the formal specifications.

5.7 Results of Questionnaires and Interviews with Project Members

A questionnaire survey was conducted towards all members of this project. At the same time, an approximately 40 minutes interview was held individually to 12 members who wrote and tested the formal specifications or referred to them frequently.

A portion of the results of the questionnaire is summarized in Table 4.

In briefly summarizing the results of the above questionnaire and interviews, we have found that there was no major adverse reaction to the application of the formal method. A commonly found opinion was that while they realized its efficiency as a communication tool, the cost for initial training was necessary.

Specifiers Generally have a favorable impression of the formal method and are positive about the effect of its application in the project. Within this project, they can confidently take on duties other than specification development.

Firmware Engineers While the specification itself is clear, it is hard to identify the boundary of the external specification and the formal specification program required execution.

Test Engineers Since the specification is clear, it is possible to test with confidence.

6 Conclusion

The application of the formal method was highly effective for the success of our project on schedule.

The formal method contributes to the quality of deliverables at the upper stream of development process.

And, the formal method appear to have encouraged communications between engineers, which is vital for software development.

Additionally, the fact that the executable formal specifications are resembled to program codes is an substantial advantage because the know-how accumulated through program development can be applied (for example,

configuration management, filter programs, batch programs, object-oriented analysis and design technology, unit testing and so on).

It is necessary to pay attention to not only executable features but also the readability of specifications. Specifications which are referred to by all project members need to be simple, so that it can be read without stress.

7 Difficulties

In our project, the capability for abstraction required by formal specification engineers did not go beyond that required by usual programmers.

In the case of VDM++, coding and testing formal specifications are not difficult for engineers who are familiar with the object-oriented design and the implementation of C++/Java languages. They would need only about a month training.

However, it is difficult to describe of the formal specifications than using general programming languages. Although a lack of VDM libraries, templates and advices from experts could be a disadvantage for beginners.

8 Future Issues

Future issues are listed below:

- Validating whether specifications fulfill requirements;
- Negotiating with stakeholders who do not read formal specifications;
- Testing user's manuals that is based on formal specifications;
- Defining effective combinations of formal and informal specifications;
- Description of formal specifications suitable for embedded systems;
- Validation and testing of the formal specifications; for example validation of whether a security specification is logically consistent;
- Framework for describing specifications that are easy-to-read and executable;
- Specifications that firmware engineers and test engineers feel close to, familiar with and comfortable browsing.

Acknowledgments

We would like to thank Professor Peter Gorm Larsen of Engineering College of Aarhus, Professor Keijiro Araki of Kyushu University, Shin Sahara of CSK Systems Corporation and Hiroshi Sako of Designers' Den Corporation for their great assistance in the application of the formal method.

References

[1] J. Fitzgerald, P.G. Larsen, P. Mukherjee, N. Plat, M. Verhoef, *Validated Designs for Object-oriented Systems*, Springer, 2005

[2] J. Fitzgerald, P.G. Larsen, *Modelling Systems: Practical Tools and Techniques in Software Development*, Cambridge University Press, 1998

[3] http://www.vdmtools.jp/en/