What Top-Level Software Engineers Tackle after Learning Formal Methods: Experiences from the Top SE Project

Fuyuki Ishikawa[1], Kenji Taguchi[1], Nobukazu Yoshioka[1], and Shinichi Honiden[1,2]

[1] GRACE Center,
National Institute of Informatics,
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan
[2] Graduate School of Information Science and Technology,
The University of Tokyo,
7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan

Abstract. In order to make practical use of formal methods, it is not sufficient for engineers to obtain general, fundamental knowledge of the methods and tools. Actually, it is also necessary for them to carefully consider their own contexts and determine adequate approaches to their own problems. Specifically, engineers need to choose adequate methods and tools, determine their usage strategies, and even customize or extend them for their effective and efficient use. Regarding the point, this paper reports and discusses experiences on education of formal methods in the Top SE program targeting software engineers in the industry. The program involves education of a variety of scientific methods and tools with group exercises on practical problems, allowing students to compare different approaches while understanding common principles. In addition, the program involves graduation studies where each student identifies and tackles their own problems. Statistics on problem settings in the graduation studies provide interesting insights into what top-level engineers tackles *after* learning formal methods.

1 Introduction

Formal methods are attracting increasing attentions from the software industry, as software is getting more and more complex while efficiency and reliability of software development is getting more and more significant. On the other hand, there has been a gap between knowledge and techniques that software engineers in the industry generally have and those required for use of formal methods. It is thus essential to provide a place where engineers in the industry can learn formal methods, expecting their practical application.

When engineers apply formal methods to problems in practical system development, they need to tackle the difficulty in determining adequate approaches to their own problems, carefully considering their own contexts. Specifically, they need to choose adequate methods and tools, determine their usage strategies, and

J. Gibbons and J.N. Oliveira (Eds.): TFM 2009, LNCS 5846, pp. 57–71, 2009.

even customize or extend them for their effective and efficient use. Therefore, education of general, fundamental knowledge and techniques of formal methods and tools is not sufficient by itself.

Regarding this point, the Top SE program in Japan has provided a unique place to produce "superarchitects" who can promote practical use of advanced, scientific methods and tools, including formal methods, for tackling problems in software engineering [1,2]. The Top SE program primarily targets engineers in the industry and provides education on a variety of methods and tools by combining lecturers from the academia and the industry. Considering the points discussed above, the program provides more than education of general, fundamental knowledge and techniques on methods and tools, in the following two forms of activities.

Lecture Courses. In the Top SE project, lecture courses are organized to involve different methods and tools so that students can compare different approaches while understanding common principles. Each course involves group exercises where students jointly tackle practical problems while exchanging ideas on different approaches to the problems. The problems are jointly developed by lecturers from the academia and the industry.

Graduation Studies. The Top SE program also includes graduation studies where students identify and tackle their own problems using scientific approaches they have learned. The studies are finally evaluated in terms of validity of problem setting, validity of approach (method/tool) selection, and problem-solving ability (e.g., adequate abstraction).

The Top SE project successfully completed its setup phase of five years with government sponsorship, through which 61 students have graduated and 21 lecture courses have been developed. In April 2009, it started a renewed phase based on the established education system, and welcomed 31 new students, out of which 29 are engineers from the industry.

This paper reports and discusses the experiences on formal methods education based on the principles described above in the Top SE project. Besides design of lecture courses, this paper focuses on the graduation studies. The statistics on graduation studies will clarify what kinds of issues the students (actually top-level engineers in the industry) find significant and also solvable to some extent by themselves. It will give interesting suggestions regarding roles of the academia and the industry for promotion of widespread, practical use of formal methods.

The remainder of the paper is organized as follows. Section 2 describes the principles and current status of the Top SE program. Section 3 describes lecture course designs on formal methods in the program. Section 4 reports actual topics of graduation studies on formal methods tackled by the students. Section 5 gives and discusses statistics obtained in the experiences, and Section 6 finally concludes the paper.

2 Top SE Program

2.1 Principles

In the software engineering area, there has been a gap between what are taught in the academia and what are required by the industry [3]. The Top SE program in Japan is organized to bridge the industry-academia gap by providing a place where the academic and the industry jointly deliver knowledge and techniques for practical use of advanced scientific methods and tools [1,2]. Figure 1 illustrates the principles of the Top SE program, which involves digital home appliances as one of the target practical areas.

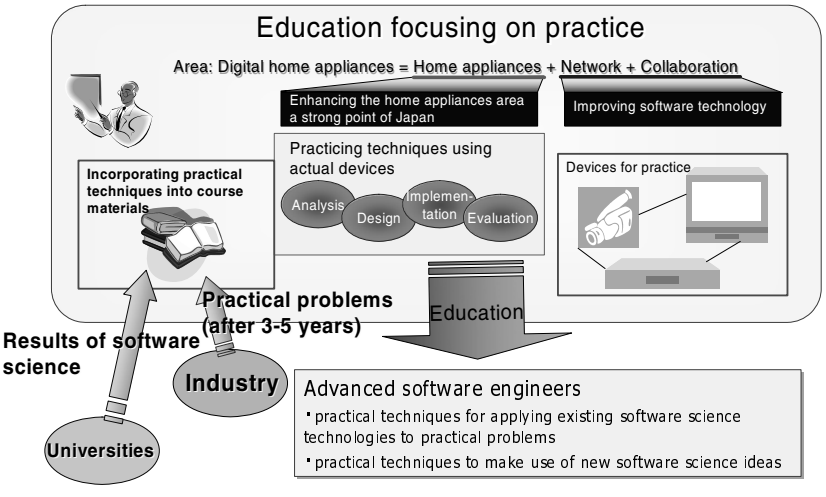

Fig. 1. Approach to Educate SuperArchitects, with the Example of Digital Home Appliances

Below summarizes principles of the Top SE program.

Target Topics. Topics cover software engineering, especially focusing upper stream processes where scientific methods and tools work effectively and efficiently. As essential goals in software engineering, efficiency, reliability, and changeability are investigated. In addition, security is also focused on as it is now an essential aspect in a variety of systems. Either hardware technologies or social matters are not considered as primary topics.

Target Students. As the original motivation is to deliver scientific approaches to the industry, the primary target students are software engineers from the industry. They know issues in software development well but do not know so much about use of scientific methods and tools. Especially, engineers around their 30's are targeted as they have identified issues in software engineering and are leading next-generation development processes. The program also

accepts graduation students of universities, who know scientific aspects well and are eager to learn their application in practical software development. For education of university students, the Top SE program also provides lecture courses at universities, which puts more focuses on delivering practical aspects to graduate students. This is out of the scope of this paper.

Produced Graduates. The project defines "superarchitects" to produce, who have knowledge and techniques of the following.

- Abstraction of practical problems into (semi) formal models, e.g., UML, automata, formal specifications, and goal-oriented requirements models.
- Application of tools to concrete problems, e.g., digital, networked home appliances.
- Adaptability to new technology and tools, e.g., identifying essential differences and common principles in similar but different approaches.
- Ability to promote the technologies and tools, e.g., instructing development teams.

Lecturers and Lecture Courses. Figure 2.1 illustrates the approach of the Top SE program to lecture course organization. Lecturers from the academia and the industry are grouped and develop lecture course materials, which involve results of software science from the academia combined with practical problem (exercise) settings from the industry. Lecture courses are organized to involve different methods and tools so that students can compare different approaches while understanding common principles. In addition, each course involves group exercises where students jointly tackle practical problems while exchanging ideas on different approaches to the problems even if using the same method and tool.

Graduation Studies. The students finally tackle graduation studies, for a few months, where they identify and tackle their own problems using scientific approaches they have learned. Students often determine to choose and apply some methods and tools for their own problem areas, e.g., workflow management system. Some students develop extension of existing methods and tools so that the methods and tools are used more effectively and efficiently for a certain class of problems, e.g., development of a model generator from a specific format. Graduation studies are evaluated in terms of validity of problem setting, validity of approach (method/tool) selection, and problem-solving ability (e.g., adequate abstraction).

Detailed description and discussion on the principles of the Top SE project are found in [1].

2.2 Current Status

The Top SE project had been fully sponsored by the Japanese government in its setup phase of 5 years until March 2009. During the phase, it accepted students three times (in 2005, 2006, and 2007) and provided them an education program of one year and half (for free). The number of the students and the number of the lecture courses were gradually increased. The third-term students (from

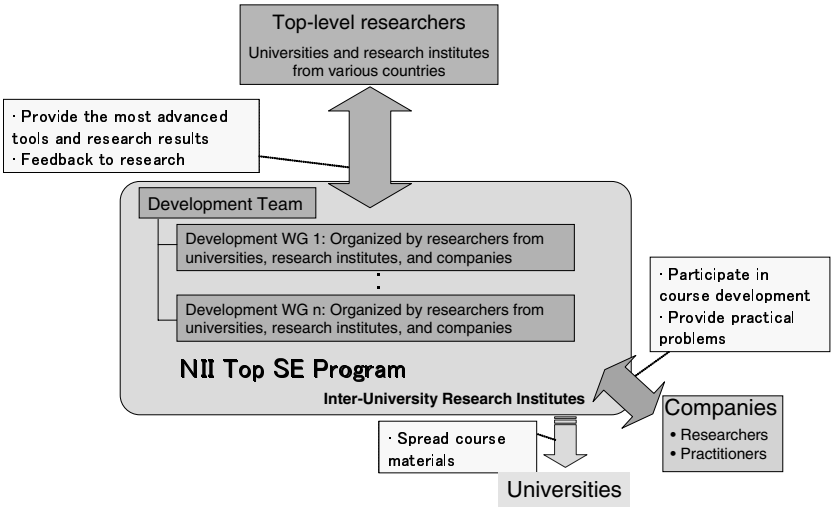

Fig. 2. Approach to Organize and Develop Lecture Courses

September 2007 to March 2009) enjoyed the educational system developed so far, which is planned to be maintained and refined continuously.

Below describes the status of the Top SE project at the end of the setup phase (March 2009).

Students. In March 2009, 30 students graduated. 4 of them were graduate students (of universities) and the others were software engineers from the industry. 61 students graduated in total in the setup phase.

Lectures. 21 lecture courses, classified in 6 series plus introductory courses, were finally developed and provided for the third-term students as shown in Table 1. Each lecture course involved 12 classes, each of which was held for one hour and a half. Classes were held in the evening (16:30-18:00, 18:15-19:45) on weekdays. Lecture courses were developed and operated by 25 lecturers, 15 from the academia (universities or research institutes) and 10 from the industry (companies). Each student must get through at least 8 courses with credit for graduation.

In addition, postgraduate support has been established on the basis of a partnership with a graduate university. A graduate of the Top SE program can proceed to doctoral course with advanced standing as well as continuous support of the supervisor of the graduation study. In April 2009, 7 of the graduates entered the doctoral course of the partner graduate university.

After successful completion of the setup phase fully sponsored by the government, the renewed Top SE project started in April 2009. In response to feedbacks obtained in the setup phase, the educational system was renewed, e.g., the start time of the lectures was moved to later in the evening. Although it became a

Table 1. Lecture Courses in the Top SE Program

Series	Course
Foundations	Fundamental Theories
	Practice of Software Engineering
Architecture	Component-based Development
	Software Patterns
	Aspect-Oriented Development
Formal Specification	Formal Specifications (Foundations)
	Formal Specifications (Applications)
	Formal Specifications (Security)
Model Checking	Verification of Design Models (Foundations)
	Verification of Design Models (Applications)
	Verification of Performance Models
	Modeling and Verification of Concurrent Systems
Requirements Analysis	Goal-Oriented Requirements Analysis
	Requirements Elicitation and Identification
	Security Requirements Analysis
	Early Requirements Analysis
Implementation Techniques	Testing
	Program Analysis
	Verification of Implementation Models
Management	Software Metrics
	Software Development Management

fare-paying education with some scholarship, 31 students joined and started to study (almost the same number as that of the third-term students).

This paper discusses formal methods education in the Top SE program, on the basis of experiences in the setup phase. Section 3 describes design of lecture courses. Specifically, among the courses enumerated in Table 1, the Formal Specification series and the Model-based Verification series are discussed as they primarily deal with formal methods. In addition, part of the Implementation Techniques series is discussed, which includes formal methods targeting source codes. Section 4 describes graduation studies on formal methods. Section 5 discusses detailed statistics on lecture courses and graduation studies.

3 Lecture Courses on FM

3.1 Model Checking Series

The Model Checking series focuses on mathematical modeling of software behaviors and their efficient verification with automated tools. According to the principles described in 2.1, The series consists of four lecture courses as illustrated in Figure 3.

MC1: Verification of Design Models (Foundations). This course delivers fundamental knowledge and techniques for use of model checking methods.

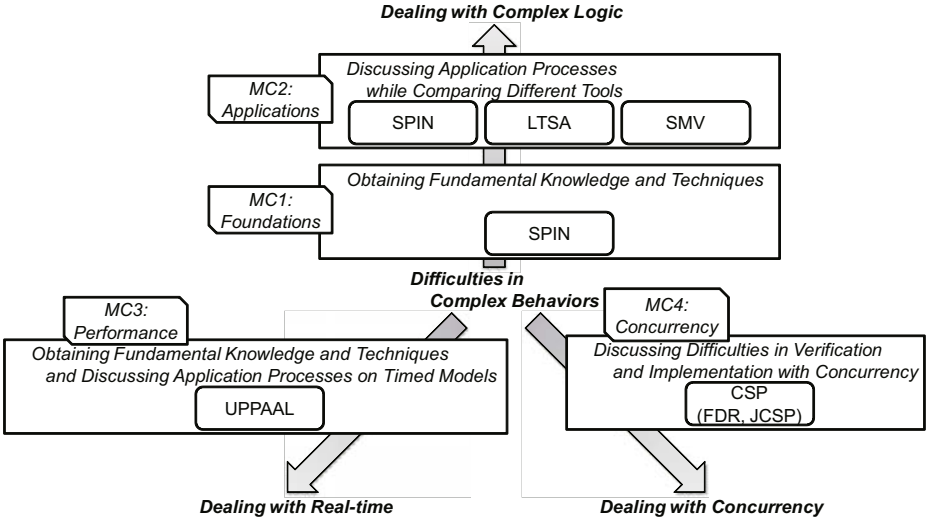

Fig. 3. Model Checking Series

SPIN [4] is used as one of the most sophisticated tools. Besides basic notations in Promela and usage of SPIN, the course explicitly focuses on the verification process, including construction of models from the design, identification of property descriptions, and validation of the verification activity itself. This process is organized systematically, involving UML modeling and tool-independent model design. For group exercise on a practical problem, coordination of networked home appliances is investigated with incorporation of actual devices. The problem setting is based on emerging practical situation to be tackled actively by Japanese companies.

MC2: Verification of Design Models (Applications). This course delivers advanced knowledge and techniques for application of model checking methods. SPIN [4], LTSA [5] and SMV/NuSMV [6] are used to tackle the same problem so that students can discuss what are common and what are different in those tools. The course also puts more focus on the problems of adequate abstraction considering the verification purpose as well as careful modeling of external environments that can lead to unexpected behaviors. For group exercise, each group defines a problem such as cellular phone control, and investigates it by discussing which tool(s) to use.

MC3: Verification of Performance Models. This course delivers knowledge and techniques for use of model checking methods for real-time systems, considering the performance aspect. UPPAAL [7] is used as one of the most sophisticated tools. The course also discusses guidelines and pitfalls in modeling timing aspects. As a practical problem, an audio-control protocol with bus collision is investigated where timing of voltage changes is essential for correct communication.

MC4: Modeling and Verification of Concurrent Systems. This course delivers knowledge and techniques for using formal models for verification and implementation of concurrent systems. CSP is used as a formalism to model concurrency, with a tool for verification of equivalence and refinement relationships, FDR2 [8], as well as a tool for deriving implementation, JCSP [9]. As difficulties in concurrent systems are so common, students define their own practical problem settings for the group exercise in this course, such as cellular phone protocols and voting protocols.

3.2 Formal Specification Series

The Formal Specification series focuses on mathematical modeling of system states and their changes through operation invocations. According to the principles described in 2.1, the series consists of three lecture courses as illustrated in Figure 4.

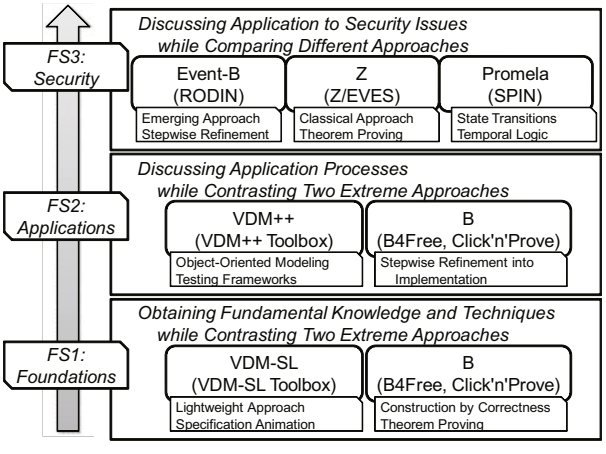

Fig. 4. Formal Specification Series

FS1: Formal Specification (Foundations). This course delivers fundamental knowledge and techniques for use of formal specification. Two different methods are discussed to deliver common principles and different decisions: B-Method aiming at correct derivation of implementation based on theorem proving and VDM aiming at lightweight validation based on specification animation and testing. As tools, B4Free and Click'n'Prove [10] are used for B-Method and VDM-SL Toolbox [11] for VDM. As a practical problem, a standard routing protocol for ad-hoc networks is investigated, which is an emerging problem with clear specifications in natural languages. Here B-Method is used to gradually introduce complexity by modeling distribution of data among nodes through stepwise refinement, while VDM is used to model behaviors of each node.

FS2: Formal Specification (Applications). This course delivers advanced knowledge and techniques for application of formal specification. B-Method and VDM are used similarly to **FS1**, but more focus is put on different levels of abstraction well as stepwise refinement through them. As tools, B4Free and Click'n'Prove are used for B-Method and VDM++ Toolbox [11] for VDM. As a practical problem, other aspects of the routing protocol for ad-hoc networks is investigated.

FS3: Formal Specification (Security). This course delivers knowledge and techniques for using formal specifications in development of secure systems. Three different methods and tools are discussed with the same practical problem of access control in a hospital. The first is Event-B [12] with the RODIN tool, where event-based modeling and stepwise refinement are investigated. The second is the Z/EVES tool [13], where theorem proving is investigated. The last is SPIN [4], where state transition-based modeling and temporal properties are investigated.

3.3 Implementation Techniques Series

The Implementation Techniques series complement focuses on verification tasks during and after implementation. Regarding formal methods, it complement with the above two series by providing lecture courses on methods and tools targeting source codes.

IM1: Program Analysis. This course delivers fundamental knowledge and techniques for analysis of programs. JML [14] is used and discussed from the view points of a means for Design by Contract, a means for unit testing, and a means for static verification of source codes. As practical problems, an access control system at a hospital and an on-line shopping system are investigated.

IM2: Verification of Implementation Models. This course delivers advanced knowledge and techniques for application of model checking to verification of source codes. Java PathFinder [15] is used and discussed. As a practical problem, network protocols are investigated. More detailed description and discussion on the course is found in [16].

4 Graduation Studies on FM

As described in 2.1, students identify and tackle their own problems in their graduation studies. Many students have chosen topics on formal methods. This paper classifies the topics as follows.

Case Study. In this type of study, problems are identified in development of some target systems, such as workflow management system and web application. Solutions are defined by choosing adequate formal methods/tools as well as defining application strategies. Below describes a few examples of this type of graduation studies.

- One of the studies used VDM for formal modeling and validation in an experimental project involving a team at the company with which the student works. It evaluated additional costs (man-hours) as well as specification items additionally identified through the modeling and validation, which would lead to much more cost if found in later phases.
- Another study used SPIN and NuSMV, respectively, for verification of access control policies on shared file operations, and detected situations where a high-level policy of confidentiality or availability is broken (not correctly implemented) by low-level policies.

Domain-Specific Finer-Grained Support. In this type of study, problems are identified in application of formal methods/tools to development of systems in some domains, such as workflow management system and web application. Solutions are defined by developing domain-specific methods/tools that provide finer-grained support for application of general formal methods/tools. Below describes a few examples of this type of graduation studies.

- One of the studies investigated translation rules from BPMN [17], a standard notation of business processes, to the notation of timed automata in UPPAAL. After adding timing constraints to a BPMN process, UPPAAL automata are obtained according to the translation rules. The study had a case study based on a simplified version of examples in the BPMN specification, where introduction of timing constraints makes it difficult to achieve consistency.
- Another study investigated generation of models from configurations in Web applications, namely, configuration files in Struts and JSP files [18].

Bridging Gaps between Different Methods/Tools. In this type of study, problems are identified in bridging between a task where formal methods/tools are used and another task where other (formal or not formal) methods/tools are used. Solutions are defined by developing methods/tools for combining formal methods/tools with other (formal or not formal) methods/tools. Below describes a few examples of this type of graduation studies.

- One of the studies investigated a method and tool that supports deriving specifications in VDM from requirements obtained by using KAOS [19,20].
- Another study investigated a method that derives properties to be verified by model checking from goal-oriented requirements models [21].

Extension of Methods/Tools. In this type of study, problems are identified in capabilities of existing formal methods/tools themselves regarding their own purposes (e.g., verification). Solutions are defined by developing extension of formal methods/tools. Below describes a few examples of this type of graduation studies.

- One of the studies defined refinement relationships in VDM++ as well as proof obligations so that specific kinds of refinement relationships, which have been recently dealt with in Event-B, can be explicitly modeled in VDM.
- Another study developed an Eclipse plug-in for SPIN, including an editor with auto completion, a wizard to specify LTL formula and invocation of the model checker.

5 Statistics and Discussion

In the following, attendance and completion at lecture courses on formal methods is first discussed. Problem settings in the graduation studies on formal methods are then discussed.

5.1 Attendance and Completion at Lecture Courses on FM

Table 5.1 shows how many students successfully got through each course with credit. It means the number of students who registered to the course, attended the classes, and completed reports on personal exercises and group exercises. In the table, the number of students who attended each course is also shown between parentheses (students who registered but not completed or who audited). Here only 30 students who graduated in March 2009 are counted, as previous graduates did not have the full lineup of 21 lecture courses described in Table 1.

Table 2. Completion and Attendance at Lecture Courses on Formal Methods

Series	Course	Students: completed (attended)
Model Checking	MC1	17 (21)
	MC2	12 (15)
	MC3	5 (10)
	MC4	8 (10)
Formal Specification	FS1	20 (27)
	FS2	14 (20)
	FS3	4 (5)
Implementation Techniques	IM1	6 (14)
	IM2	5 (6)

(Students in total: 30)

It would be difficult to discuss valid meanings of the figures, because there may be different kinds of reasons why students did not attend courses and why they did not complete. For example, it is often the case that a student wants to attend a course, but it is held on the day of the week when he/she has to stay at his/her company. In such a case, students often just download the slides and other materials and try to study by themselves (asking the lecturers questions if necessary). Another aspect is that **MC4** and **FS3** were held later in the program and most of the students had completed sufficient number of courses and wanted to focus on their graduation studies.

Anyway, it seems that most of the students were interested in **MC1** and **FS1**, that is, introductory courses on formal methods.

5.2 Selection of FM in Graduate Studies

28 studies were related to formal methods, out of 61 studies. Here 2 studies are not counted that mentioned some formal methods while discussing whole

pictures of development processes or educational programs to be used in the students' companies. The result shows that about half of the graduation studies used formal methods. In addition, 5 students determined to continue investigation of formal methods by getting into doctoral course of the partner graduate university. These results suggest formal methods are somewhat popular among educated top-level engineers in Japan.

5.3 Tool Selection in Graduate Studies on FM

Table 5.3 shows the number of graduation studies, on formal methods, that use each tool.

Table 3. Tools in Graduate Studies on Formal Methods

Series	Tool	Number of Studies
Model Checking	SPIN	8
	UPPAAL	2
	CSP (FDR/JCSP)	3
	Tool-Independent	1
Formal Specification	VDM	5
	Event-B	3
Implementation Techniques	JML (ESC/Java2)	1
	Java PathFinder	1
Combination	SPIN and SMV/NuSMV	1
	SPIN and Java PathFinder	1
	VDM and SPIN	1
	VDM and Event-B	1

(28 Studies on FM, out of 61)

The one classified as "Tool-Independent" is investigation of the application process of model checking. It defined roles of involved engineers and inputs/outputs exchanged between them, considering expertise required to use model checking.

Model checking was quite popular in the graduate studies. Especially, SPIN was used by many studies. It would be because students were much more familiar with SPIN as they had used it primarily (in the lecture course **MC1**).

On the other hand, VDM was also popular, which would be somewhat unique in Japan. It would be because VDM Toolbox is maintained by a Japanese company and Japanese interface has been provided. It would be also because of the well-known application case of VDM for an IC card system, which is used extensively in Japanese people's life [22]. Students who chose VDM tended to claim that they would not be able to introduce formal methods heavier than VDM, the lightweight one, to their colleagues and companies.

5.4 Topics in Graduation Studies on FM

Table 5.4 shows the number of graduation studies, on formal methods, that belong to each classification described in Section 4.

Table 4. Topics in Graduate Studies on Formal Methods

Classification	Number of Studies
Case Study	6
Domain-Specific Finer-Grained Support	11
Bridging Gaps between Different Methods/Tools	7
Extension of Methods/Tools	4

(28 Studies on FM, out of 61)

Among the studies on formal methods, Domain-Specific Support was investigated most actively. Motivations discussed in the studies suggest two points of the following.

- When students discusses with themselves how they can solve their problems in software development in their domains, by using formal methods, they often find other problems appear in applying formal methods.
- When students discusses with themselves how they can solve the problems found in applying formal methods, they often reach some initial ideas to provide methods/tools to support application of formal methods to their own domains. Although the studies are done in a few months and the initial ideas are somewhat naive, students have been able to present a core part of the ideas as well as simple case studies to show their effectiveness.

In other words, for widespread use of formal methods, domain-specific methods and tools providing finer-grained support are inevitable. Top-level engineers in the industry have sufficient abilities and motivations to provide them when they are given opportunities to do so.

Bridging Gaps between Different Methods/Tools suggests similar points. Methods and tools are inevitable that manage input to formal methods and/or output from formal methods, e.g., connecting requirements and model checking, connecting UML and formal specification, and so on.

On the other hand, it seems difficult for general engineers to extend existing methods and tools in terms of their own purposes, e.g., improving efficiency of model checkers. Actually, two of this kind of studies were simple, dealing with syntax issues (auto-completion and syntax sugars) while one essential extension was investigated by a research-oriented graduate student. The other one is investigation of the application process of model checking discussed in the previous section, which is more meaningful when deployed in the industry.

6 Conclusion

The Top SE project has been established to deliver scientific approaches in software engineering, including formal methods, to engineers in the industry. It provides lecture courses developed jointly by lecturers from the academia and from the industry, as well as opportunities of graduation studies where students identify and tackle their own problems.

This paper has reported and discussed experiences on formal methods education in the Top SE project. Students, mostly engineers in the industry, were eager to learn formal methods, both model checking and formal specification, which are used effectively and efficiently in early phases of development.

For practical use of formal methods, it is not sufficient for engineers to obtain general, fundamental knowledge of the methods and tools. It is also necessary for them to carefully consider their own contexts and determine adequate approaches to their own problems. Specifically, engineers need to choose adequate methods and tools, determine their usage strategies, and even customize or extend them for their effective and efficient use.

The Top SE project has provided educational experiences to polish this kind of abilities of engineers, through group exercises and graduation studies. Especially, experiences in graduation studies are unique, where engineers identify and tackle problems by themselves. This paper has discussed what the experiences have pointed out regarding engineers' points of view on formal methods. They consider formal methods as what are somewhat incomplete by themselves and what SHOULD be and CAN be complemented by providing domain-specific methods/tools as well as methods/tools bridging gaps between different methods/tools.

The Top SE program has established its education system that accepts about 30 students per year. After discussion given in this paper, we believe the approach of the program will promote practical use of formal methods, where not only the academia but also the industry develop practical support for effective and efficient use of formal methods.

Acknowledgments

The Top SE program is fully sponsored by the Special Coordination Fund for Promoting Science and Technology, for fostering talent in emerging research fields by MEXT, the Ministry of Education, Culture, Sports, Science and Technology, Japan. We would like to thank all the lecturers and students who have worked very hard in the Top SE program, providing the unique, interesting experiences discussed in this paper.

References

1. Honiden, S., Tahara, Y., Yoshioka, N., Taguchi, K., Washizaki, H.: Top SE: Educating Superarchitects Who Can Apply Software Engineering Tools to Practical Development in Japan. In: The 29th International Conference on Software Engineering, pp. 708–718 (2007)
2. Top SE project (NII), `http://www.topse.jp/`
3. Beckman, K., Coulter, N., Khajenoori, S., Mead, N.R.: Collaborations: Closing the industry-academia gap. IEEE Software 14(6), 49–57 (1997)
4. SPIN - formal verification, `http://spinroot.com/`
5. LTSA - Labelled Transition System Analyser, `http://www.doc.ic.ac.uk/ltsa/`

6. The SMV System, http://www.cs.cmu.edu/~modelcheck/smv.html
7. UPPAAL, http://www.uppaal.com/
8. Formal Systems (Europe) Ltd., http://www.fsel.com/
9. Communicating Sequential Processes for Java (JCSP), http://www.cs.kent.ac.uk/projects/ofa/jcsp/
10. B Method - Presentation of B Method, B Language, and formal methods, http://www.bmethod.com/
11. VDM information web site, http://www.vdmtools.jp/
12. Rodin - rigorous open development environment for complex systems, http://rodin.cs.ncl.ac.uk/
13. Saaltink, M.: The Z/EVES System. In: Till, D., Bowen, J.P., Hinchey, M.G. (eds.) ZUM 1997. LNCS, vol. 1212, pp. 72–85. Springer, Heidelberg (1997)
14. The Java Modeling Language (JML), http://www.cs.ucf.edu/~leavens/JML/
15. Java PathFinder, http://javapathfinder.sourceforge.net/ (last Access, April 2008)
16. Artho, C., Taguchi, K., Tahara, Y., Honiden, S., Tanabe, Y.: Teaching software model checking. In: Formal Methods in Computer Science Education, FORMED 2008 (2008)
17. BPMN information home, http://www.bpmn.org/
18. Kubo, A., Washizaki, H., Fukazawa, Y.: Automatic extraction and verification of page transitions in a web application. In: The 14th Asia-Pacific Software Engineering Conference, ASPEC 2007 (2007)
19. Goal-Driven Requirements Engineering: The KAOS Approach, http://www.info.ucl.ac.be/~avl/ReqEng.html
20. Nakagawa, H., Taguchi, K., Honiden, S.: Formal specification generator for KAOS: Model transformation approach to generate formal specifications from KAOS requirements models. In: The 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007), pp. 531–532 (2007)
21. Ogawa, H., Kumeno, F., Honiden, S.: Model checking process with goal oriented requirements analysis. In: The 15th Asia-Pacific Software Engineering Conference (ASPEC 2008), pp. 377–384 (2008)
22. Kurita, T., Chiba, M., Nakatsugawa, Y.: Application of a Formal Specification Language in the Development of the "Mobile FeliCa" IC Chip Firmware for Embedding in Mobile Phone. In: Cuellar, J., Maibaum, T., Sere, K. (eds.) FM 2008. LNCS, vol. 5014, pp. 425–429. Springer, Heidelberg (2008)