Keys and Roles of Formal Methods Education for Industry: 10 Year Experience with Top SE Program

Fuyuki Ishikawa[1], Nobukazu Yoshioka[1], and Yoshinori Tanabe[1,2]

[1] National Institute of Informatics, Japan
[2] Tsurumi University, Japan
f-ishikawa@nii.ac.jp

Abstract. Formal methods play essential roles in education and training of software engineering. Besides the fact that formal methods themselves can be direct solutions to problems in development, many insights into modelling, specification, verification and validation can be provided. On the other hand, difficulties lie in delivering these direct or indirect values to the industry. This paper reports and discusses our experience with formal methods education in the Top SE program. This program has been provided for industry engineers to learn advanced software engineering. The program contains over 10 lectures on formal methods, from theoretical foundations to practice including non-technical aspects. The program also allows the engineers to spend 3 or 6 month tackling their own problems with supervisors. Our experience in the program illustrates key factors and roles of formal methods education.

1 Introduction

Formal methods are attracting increasing attention, as solutions to the high demand of efficient and reliable software development. On the other hand, there has been a gap between knowledge and techniques that software engineers generally have obtained and those that are required for use of formal methods. It is thus essential to provide a place where software engineers can learn formal methods and investigate their practical applications.

Formal methods can also be referred to as foundations that illustrate a set of principles and rationales in software engineering. For example, modelling vocabularies and structures in formal languages can suggest good practices in specification, rigorously defining “what” (not “how”), even if encoded in natural languages. This point also suggests the significance of formal methods education in the general contexts of software engineering.

This paper reports and discusses our experience of formal methods education with the Top SE program [2,7]. The program started about 10 years ago (2004), and has provided a unique place for education of advanced techniques of software engineering to the industry. This education includes not only over 40 lectures but also so-called graduation studies. In these graduation studies, the engineers identify and tackle their own problems with supervisors. The number of engineers who have participated in the program is now over 300.

Among various aspects of our experience, this paper reports how the engineers studied and evaluated formal methods to discuss key factors and roles of formal methods education for the industry. This focus is different from our previous paper in 2009 [8], which reported statistics on formal methods studies.

The remainder of the paper is organized as follows. Sections 2 and 3 give an overview of the Top SE program and formal methods education there. Section 4 provides analysis and discussion on the key factors and roles of formal methods education for the industry. Section 5 gives concluding remarks.

2 Top SE Program

In the software engineering area, there has been a gap between what is taught in academia and what is required by industry [4]. The Top SE program was developed to bridge this gap by providing a place where academia and industry jointly deliver knowledge and techniques for practical use of advanced scientific methods and tools.

The Top SE program [2, 7] targets engineers in the industry and provides education on a variety of methods and tools by lecturers from academia and the industry. The Top SE program first started with a government sponsorship of 5 years [3]. After that, the program was renewed as a paid course, and has been operated very stably. As of 2015, it has each year:

- 44 lecture classes
- 40 students (mostly engineers from the industry)
- 47 lecturers (about two thirds from the industry)

Each student spends 1 or 1.5 years on lectures as well as a so-called graduation study, as explained below.

2.1 Lectures

In the Top SE program, lectures are organized to involve different methods and tools so that students can compare different approaches while understanding common principles. Each class involve group exercises in which students jointly tackle practical problems while exchanging ideas on different approaches.

Each lecture consists of 7 or 15 slots (each slot is 1.5 hour). The 44 lectures are grouped into six “lecture series,” Project Management, Requirement Engineering, Architecture, Formal Specification, Model Checking and Cloud Computing. Each lecture series consists of six to ten classes. As the students are industry engineers, classes are held in the evenings (6:20pm-) every weekday and in the daytime on Saturdays. Students are required to take at least six classes to graduate. About a quarter of them take this minimum number, but others take much more, in extreme cases more than 20.

[3] By the Ministry of Education, Culture, Sports, Science and Technology, Japan.

Although acquisition of knowledge and exercises on methods and tools are indispensable, solving each problem requires discussions on and decisions regarding approaches such as modelling scope, right abstraction, and properties for verification. For exercises for such aspects, most classes have group exercises in which students jointly tackle non-trivial problems, i.e., Problem-based Learning (PBL).

2.2 Graduation Studies

The Top SE program also includes graduation studies in which students identify and tackle their own problems using approaches they have learned. The duration of the studies is 3 or 6 months, guided by a lecturer(s) as a supervisor. The studies are finally evaluated in terms of validity of problem setting, approach (methods/tools), modelling of target problems, and evaluations.

A few of typical patterns of graduation studies are illustrated below.

Case Study In this type of study, engineers identify problems in development activities at their companies. They tackle these problems by selecting and tailoring scientific methods and tools learnt in the course, and evaluate how well the problems are solved. A typical example is application of a formal verification tool to a typical but troublesome problem and evaluation of cost and effectiveness.

Domain-Specific Finer-Grained Support In this type of study, engineers identify problems in the application of methods and tools learnt in the course. They tackle these problems by providing fine-grained support for application, often in a domain-specific way. A typical example is a translation tool from spreadsheets currently used for the specific input format of a formal verification tool.

Bridging Gaps between Different Methods/Tools In this type of study, engineers identify problems in bridging different phases or tasks, for each of which methods and tools are often discussed separately. They tackle these problems by providing support for connecting output of one task to the input of another. A typical example is support of systematic mapping from requirement models to design models.

Development of Methods and Tools In this type of study, engineers identify problems in targets or capabilities of current methods and tools. They tackle these problems by constructing new methods and tools. A typical example is tool development to support deeper analysis or visualization of counterexamples generated by a model checker.

3 Formal Methods in Top SE Program

Our previous paper reported on formal methods education at the Top SE program [8] in 2009. Notable points continuously delivered from the previous report as well as updates are summarized below.

3.1 Lectures

Intensive Courses Among the 44 classes, 13 are about formal methods, including 1 about mathematical logic.

The class for mathematical logic teaches the set theory, predicate logic, hoare-logic, temporal logic, and so on. The knowledge is given basically from the viewpoint of "users" of the theories, e.g., those who do not prove soundness.

The other 12 classes are divided in two categories, 6 for formal specification and 6 for model checking. Each category starts with a basic class that focuses on one method, VDM [6] and SPIN [1] respectively. In the basic classes, students learn the basics and receive experience in modelling, verification, and validation. At the time of writing this paper, the courses cover many methods and tools: VDM, B-Method, Event-B, Coq, SPIN, SMV, LTSA, CSP, and UPPAAL.

Group Exercises For example, group exercises for formal specification methods use a routing protocol for ad-hoc network, Optimized Link State Routing Protocol (OLSR) [3]. Each exercise focuses on specific aspects, such as an algorithm for efficient multicast avoiding duplicate deliveries and distributed management of network information. Discussions include what aspects and properties to focus on, how to abstract away details irrelevant to the focus, and so on.

Other examples of group exercises include robots of the classical "dining philosopher" problem. This is an extension of the classical problem since the robots can act in parallel, e.g., leading to a state in which two robots touch one folk at the same time. It is also possible to run actual robots by using the models. Discussions include how to define abstract and essential state transitions, what properties to verify, and so on.

Classes for Practice Two classes for practice were introduced to formal specification and model checking. The lecturers for these classes are from the industry and have application experience of formal methods.

Both classes teach principles and practice on technical or non-technical aspects, such as extraction of models from specification as well as communication and understanding within the team. The lecturers also deliver insightsdrwon from their experiences, such as the significance to explain the cost and benefit, the fact that formal methods is just one of possible means, and so on.

In the practice class for formal specification, students list questions regarding issues about specification in general and formal methods. Discussions are conducted concerning possible answers and rebuttals. The results for the first few years led to a list of over 100 FAQs. In recent years, the list has ben discussed in a shorter time and discussions moved to a new topic: decisions and planning necessary for specification (when natural languages are used and when formal specification languages are used).

In the practice class for model checking, each student plays the role as an expert of model checking, who is asked to support a project given a specification document or source code. He/she is responsible for not only checking but also presenting the work plan as well as reporting of the cost and benefit.

	Very Difficult	Difficult	Normal	Easy
Basic (FS)	10.2% (5/49)	38.8% (19/49)	46.9% (23/49)	4.1% (2/49)
Basic (MC)	7.1% (3/42)	71.4% (30/42)	19.0% (8/42)	2.4% (1/42)
Practice (FS)	13.3% (2/15)	26.7% (4/15)	53.3% (8/15)	6.7% (1/15)
Practice (MC)	0% (0/9)	33.3% (3/9)	66.7% (6/9)	0% (0/9)

Table 1. Difficulties

	Rich	Sufficient	Neutral	Few
Basic (FS)	24.5% (12/49)	36.7% (18/49)	32.7% (16/49)	6.1% (3/49)
Basic (MC)	19.4% (6/31)	25.8% (8/31)	48.4% (15/31)	6.5% (2/31)
Practice (FS)	53.3% (8/15)	26.7% (4/15)	13.3% (2/15)	6.7% (1/15)
Practice (MC)	60.0% (6/10)	40% (4/10)	0% (0/10)	0% (0/10)

Table 2. Availability of Practical Information

	Fully Applicable	Mostly Applicable	Somewhat Applicable	Not So Applicable
Basic (FS)	8.2% (4/49)	18.4% (9/49)	67.3% (33/49)	6.1% (3/49)
Basic (MC)	6.5% (2/31)	16.1% (5/31)	67.7% (21/31)	9.7% (3/31)
Practice (FS)	13.3% (2/15)	53.3% (8/15)	26.7% (4/15)	6.7% (1/15)
Practice (MC)	10% (1/10)	40% (4/10)	50% (5/10)	0% (0/10)

Table 3. Applicability

3.2 Graduation Studies

There have been many studies that constantly use formal methods. For example, there were 11 studies among 41 studies for students who entered in 2013. The best award studies, 1-3 selected each year, included at least one study on formal methods. These points are discussed in detail in the following section.

4 Analysis and Discussion

This section extracts some data from our experience in the Top SE program and discusses key factors and roles of formal methods education in the whole software engineering.

4.1 Feedbacks to Lecture Classes

Tables 1, 2 and 3 summarize the comments to the questionnaire after lectures from 2012 to 2014. Only three questions are presented and discussed here: difficulties, availability of practical information, and applicability of the obtained approaches. We used the first basic classes and the practice classes (described in Section 3.1) as the key samples for formal specification (FS) and model checking (MC). In each table, the rightmost choice is omitted since no one selected ("very easy," "no practical information" or "not applicable at all").

Difficulties Table 1 in concerned with difficulties. Although the basic classes are the starting points, many students felt they were difficult, though not too much. Many comments for both classes included difficulties due to being unfamiliar and new to logic expressions. Model checking was thought more difficult. This is probably because the problems are complex with concurrency, while the basic class for formal specification primarily focuses on sequential specification animation. The practice classes were not thought difficult since practical issues, such as management, were the main topics. A few students thought such issues were the really difficult ones, which do not have absolutely right answers and require a wide viewpoint.

Practicality Table 2 is concerned with the availability of practical information. The fundamental classes involve non-trivial group exercises and give guidelines such as modelling patterns and mapping from UML diagrams to formal models. Although some students mentioned this was very sufficient, most thought it was not. The practice classes complement this point, and most students seemed satisfied with the practice classes. Most of the positive comments suggested that the students liked the "real" nature of the exercise problems, guidance for reporting, and so on. A few negative comments on the practice class for formal specification required more details, which were difficult to provide in the limited time of the lectures.

Applicability Table 3 shows the applicability of the obtained approaches. Even though the students liked the content of the classes, many had some doubts on the immediate applicability. This point did not change much even with the satisfaction of the practice classes.

Typical directions of positive comments are given below.

- Positive comments often included specific applications and aspects, for which each commenter felt that formal methods can be (cost-)effective.
- Some comments included "what is left should be done by each project and team," suggesting that there were some gaps but the students thought the gaps cannot be bridged in a general way.
- Notable comments of non-negligible numbers were about immediate applicability of "ways of thinking," not the methods or tools themselves. For example, some included the use of principles for specification in natural languages, such as structuring of information and clarification of properties. This point suggests learning formal methods allows the learning of principles significant in handling issues of early phases. On the other hand, these types of comments imply doubts on applicability of methods or tools themselves, as suggested from negative comments.

These positive results are encouraging, as they show that the students understood and could imagine how each project and team finally find and tailor solutions for themselves, without any "silver bullets."

Typical negative comments were as follows.

- Many comments included that the languages and methods seem too special compared with the currently available practice and skills. The students could not imagine how they are disseminated and used immediately back at their companies.
- There were comments that require the methods and tools to have more useful functions. This point is true compared with popular tools such as programming IDEs.
- Some comments for the basic classes suggested that the students could not imagine concrete applications. This type of comment indicates the need for the practice classes and does not appear in the comments for the practice classes.

From the viewpoint of education, it is good to see many negative comments accompany positive comments regarding the experiences and ways of thinking the students obtained. However, the negative comments also show long-lasting issues on applicability of formal methods, e.g., as discussed in [5].

4.2 Graduation Studies

Popularity In Section 3.2, we explained the number of graduation studies as recently being about a quarter of the number of all the studies. This number is actually smaller compared with the previous report, in which studies on formal methods were about half of all studies [8]. Possible reasons include the following.

- There were strong dissemination activities, thus, strong interests in formal methods in Japan at the time of the previous report, e.g., surveys and guidelines by ministries, public agencies, and private consortia.
- There is now more variety of methods and tools taught in the program, e.g., large data processing in clouds.
- There is also more variety of engineer types, even who have little programming experience, as the program has become a standard choice in companies rather than special for limited elites.

The current ratio of studies with formal methods (a quarter) is quite fair, which is almost the same as the ratio of formal methods classes among the total number of classes.

Quality At least one study on formal methods received best awards every year. This is probably because outputs with formal methods are clearer and more repeatable and often realized as frameworks or tools with objective rationales (compared with methods that guide human thinking on requirements, risks or architectures). This point never implies any superiority of formal methods. In any case, clarification of required human efforts is significant, as well as validation of the meaning and impact of the studies.

Nevertheless, formal methods provide good opportunities for students to discuss and define approaches to be rigorous, repeatable, objective, and general.

5 Summary

We discussed the latest status of our experience with formal methods education in the Top SE program. Although it is difficult to empirically prove any direct or indirect effect of formal methods education, we have reasonable confidence in the following topics we focused on in this paper:

- Key factors in formal methods education for the industry
  - Lecturers from the industry
  - Delivering, working out, and discussing practice including non-technical aspects, such as planning and reporting.
- Roles of formal methods in software engineering education and training
  - Opportunities through exercises to discuss and define approaches to be rigorous, repeatable, objective, and general.
  - Clear principles, especially in early phases of development processes, which can be applied to processes without formal methods.

The Top SE program has been established to be sustainable as a paid course, accepting over 40 students from companies every year. We would like to have more experience and provide more insights into education as well as academia-industry collaborations.

Acknowledgements

We would like to thank all the lecturers and students (engineers) involved in the Top SE program.

References

1. SPIN - formal verification. http://spinroot.com/
2. Top SE project (NII). http://www.topse.jp/?lang=en
3. Optimized link state routing protocol (olsr). http://www.ietf.org/rfc/rfc3626.txt (2003)
4. Beckman, K., Coulter, N., Khajenoori, S., Mead, N.R.: Collaborations: Closing the industry-academia gap. IEEE Software 14(6), 49–57 (1997)
5. Bjrner, D., Havelund, K.: 40 years of formal methods - some obstacles and some possibilities? In: The 19th International Symposium on Formal Methods (FM 2014). pp. 42–61 (June 2014)
6. Fitzgerald, J., Larsen, P.G., Mukherjee, P., Plat, N., Verhoef, M.: Validated Designs for Object-oriented Systems. Springer (2005)
7. 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 (ICSE 2007). pp. 708–718 (2007)
8. Ishikawa, F., Taguchi, K., Yoshioka, N., Honiden, S.: What Top-Level Software Engineers Tackles after Learning Formal Methods - Experiences from the Top SE Project. In: The 2nd International FME Conference on Teaching Formal Methods (TFM 2009). pp. 57–71 (November 2009)