Exploring the Role of Logic and Formal Methods in Information Systems Education

Anna Zamansky[1,2](✉) and Eitan Farchi[1,2]

[1] University of Haifa, Haifa, Israel
annaz@is.haifa.ac.il

[2] IBM Research Labs, Haifa, Israel
farchi@il.ibm.com

Abstract. This position paper contributes to the ongoing debate on the role played by logic and formal methods courses in the computing curricula. We report on an exploratory empirical study investigating the perceptions of Information Systems students on the benefits of a completed course on logic and formal specification. Participants indicated that the course had fostered their analytical thinking abilities and provided them with tools to handle abstraction and decomposition. This provides a starting point for a discourse on the benefits of formal methods courses for IS practitioners.

1 Introduction

Several ways in which formal methods should be incorporated into the computing curricula have been proposed (see, e.g., [1,10,12–16,18,19]), but there is still no consensus on what and how to teach. Moreover, there is an ongoing debate on the role of the very foundations of formal methods - logic and discrete mathematics in CS education. While the early CS curricula were strongly mathematically oriented, many voices are recently calling for a less mathematically rigorous curriculum, claiming this type of knowledge is not really used by practitioners in industry [2,8]). In the Information Systems discipline, which draws a large portion of its body of knowledge from CS, the importance of logic and formal methods is even less recognized. Indeed, the ACM IS curriculum guidelines [17] mention statistics and probability as required core IS topics and discrete mathematics only as an optional one, leaving logic and formal methods outside mainstream IS topics.

We believe that this state of affairs is most unfortunate. Although we tend to agree that a typical IS major may need a less extensive mathematical background than a CS major, formal methods are an essential tool for software quality control, i.e., activities for checking (by proof, analysis or testing) that a software system meets specifications and that it fulfills its intended purpose. But - even more importantly - learning formal methods fosters analytical thinking, provides the tools to deal with abstractions, and makes the students comfortable with complex mathematical notations. As stated by J. Wing in [19]: "Thinking

D. Bianculli et al. (Eds.): SEFM 2015 Workshops, LNCS 9509, pp. 68–74, 2015.
DOI: 10.1007/978-3-662-49224-6_7

in terms of formal methods concepts forces the designer to take a more abstract perspective of a system than that taken with an algorithmic or operational approach. This more abstract thinking invariably provides the designer with new insights and a deeper understanding of the systems desired behavior".

As convincing as this may sound, as noted by Wing, "the biggest obstacle is getting "buy-in" from our colleagues: convincing co-instructors, curricula committees and administrators that integrating formal methods is a good thing to do". This requires collecting empirical evidence on the above mentioned benefits of studying formal methods. In this position paper we take a step towards filling this gap by reporting on our findings collected when teaching a graduate course "Logic and Formal Specification", taught at the Information Systems Department of the University of Haifa. The details of the course design and implementation are provided in [21]. Here we focus on the human factor, namely the students' perceptions of the benefits of the course. We present the results of our exploratory study involving 22 participants who completed our course. The students were asked about their perception of its benefits, both for an IS practitioner in general, and for them personally. The most striking observation emerging from our data analysis was that most of the students reported an effect on their *cognitive* processes: fostering analytical thinking, mental decomposition of complex problems into simpler ones, and using abstraction. This provides a starting point for collecting further empirical data on the benefits of teaching formal methods to IS practitioners.

2 Related Work

Several works address the relevance of discrete mathematics, logic and formal methods for practitioners, mainly in the context of Computer Science and Software Engineering (to the best of our knowledge, no work addressed the IS domain in this context). In [9] the suitability of the standard logic syllabus to the needs of CS practitioners is questioned: "The current syllabus is often justified more by the traditional narrative than by the practitioners needs... The proof of the Completeness Theorem is a waste of time at the expense of teaching more the important skills of understanding the manipulation and meaning of formulas". According to [9], the needs of CS practitioners are to: (i) understand the meaning and implications of modeling the environment as precise mathematical objects and relations; (ii) understand and be able to distinguish intended properties of this modeling and side-effects; (iii) be able to discern different level of abstraction, and (iv) understand what it means to prove properties of modeled objects.

In [18,19], J. Wing stresses the importance of integrating formal methods into the existing CS curriculum by teaching their common conceptual elements, including state machines, invariants, abstraction, composition, induction, specification and verification. She states discrete mathematics and mathematical logic as crucial prerequisites. Further concrete proposals on the integration of formal methods into CS curriculum are made in [1,10,12–16]. Some of these studies include some form of empirical evaluation. Their methodology is mainly based on objective assessment, comparing the performance of some control group to other groups of students with respect to programming skills [10] and general problem-solving skills, including using abstraction [13,14]. However, empirical evidence for the benefits of formal methods courses is still very sparse.

In this paper we take a different approach to providing such evidence. Instead of directly assessing the students' skills and abilities, we turn to them for help with our investigation. After all, investigating the students' perceptions may provide new insights into the way in which taking the course affected their cognitive processes. Moreover, their attitudes towards the practical value of such courses may be helpful with the prediction of their future acceptance of formal methods in industry. It should be noted that we apply a qualitative research approach using open-ended questionnaires, as opposed to the quantitative approach taken in [8], which also surveyed "what subject matter practitioners themselves actually find most important in their work", but used a closed-ended questionnaire, with questions such as "How useful have the details of this specific material been to you in your career as a software developer?" using a scale from 0 to 5, leaving no place for exploring cognitive aspects affected by the taught courses.

3 The Exploratory Study of Students Perceptions

The course "Logic and Formal Specification" has been taught at the IS department at the University of Haifa for several years by both of the authors[1]. The course is a mandatory course for graduate students, and its length is one semester, 4 h per week. Many of the students return to their studies after several years in the industry. This poses a twofold challenge when designing a course in logic and formal methods. First, they have a solid understanding of topics that have direct relevance to practice and are reluctant to study topics whose relevance to their daily practice is indirect. Secondly, they have forgotten the basic concepts of discrete mathematics which they studied years ago.

Our course design is based on previous proposals on the adaptation of the traditional logic and formal methods syllabi to the needs of modern practitioners [4,9,10,16,19]. As such, the course aims to equip the students with the following abilities: (1) read, write and understand formal specifications, (2) be able to formalize informal specifications, (3) analyze specifications and detect sources of incompleteness, inconsistency and complexity, (4) reason about specifications, and (5) check a system against a specification. Based on the above, the taught material includes (a) Basic principles for reasoning about sets; (b) Induction

[1] Perhaps it is important to mention here the authors' relevant background. The first author is an associate professor at the Information Systems Department at the University of Haifa with active research interests in applied logic. The second author is the manager of the Software Performance and Quality research group at the IBM Haifa Research Laboratory, and a member of the IBM corporate Board of Software Quality. Both of the authors have several years of experience in teaching logic and formal methods to various audiences of students.

and invariants; (c) Propositional and first-order logic and (d) Formal specification using the Z language. Further details about the course and the ways we propose to overcome the above mentioned challenges with respect to the target audience are provided in [21].

In what follows we describe the results of an exploratory interview we carried out in orderto gain a deeper understanding into the students' perception of the benefits of studying formal methods for IS practitioners. For this purpose, we chose an open-ended questionnaire [11] over indepth interviews to ensure the anonymity of our participants, which was an important concern in our context. We used an open-ended questionnaire as we wanted to minimize any presuppositions on the participants' responses (as opposed to e.g., the closed-ended questionnaire of [8]).

The answers were collected by the first author from twenty two graduate students who completed the course in the years 2013–2014. This sample included 8 female and 15 male students; 12 students out of 22 had no prior experience in industry. The questionnaire included the following open-ended questions.

Q1. Is it important for practitioners whose work is related to software development to study logic and formal methods? Why?
Q2. In what way (if at all) is the course's content useful for <u>Information Systems</u> practitioners?
Q3. What (if at all) were the course's contributions for you personally?
Q4. How relevant was your background from Discrete Mathematics course? In what way (if at all) was it helpful?
Q5. In what ways would you recommend to improve the course?

In what follows we refer mainly to the answers received to questions Q2 and Q3. Only three students responded that logic and formal methods are not useful (Q2):

1. I worked at two different places in industry, and never have I seen the courses' content put to any use...
2. It is not necessary for software development.
3. It depends on the work environment. I think it's not useful.

Two of them thought the course was not useful for them personally (Q3).

Out of those who responded positively to both questions, one of the most striking observations was the extensive use of formulations related to mental processes, such as "thinking", in particular "analytical/logical thinking" in answers to both questions.

E.g., answers to question Q2 included:

1. It improves **thinking** about problem modeling.
2. I think that it opens directions for **thinking** about how things really work under the surface.
3. The world of software is based on understanding the needs and modeling them in precise terms. Many such models require **logical thinking**.

4. The course's contents develop and deepen **ways of thinking**.
5. The course helps shaping **thinking** that can help in programming.
6. The course improves **analytical thinking**.
7. The course is very helpful in improving **thinking that is not necessarily algorithmic**. A different one, out of the box.
8. Of course! **Correct and systematic thinking** of IS practitioners helps in requirements specification.

Notably, no participants provided concrete examples of direct use of the courses' content in answering Q2. Yet several of them took a confident stand when speaking of their own personal experience in Q3:

1. I have already applied the new skills at work, using truth tables and proofs.
2. It improved my modeling skills. I'm certain!
3. I am now using the tools when reading scientific papers.
4. I was surprised to see how helpful the tools we studied are in practice.

Moreover, when answering question Q3, several participants referred again (implicitly or explicitly) to an improvement in their *mental* processes:

1. The course introduced order into complex topics. It gave me tools to simplify complex problems and find easy and efficient solutions.
2. It made me think in a modular way, providing me with the ability to grasp more complex models.
3. It improved my ability to refer to problems schematically.
4. It provided me with an abstract view on the problems of software design.
5. It made me realize there are systematic solutions to problems that seem unsolvable at first.
6. I learned to reduce complex problems to simpler ones.

Table 1 summarizes the main skill categories that emerged during text analysis of questions Q2 and Q3, providing the number of students that used formulations related to these categories.

Table 1. Categories emerging from answers to Q2 and Q3 and number of students using each category

	Q2 (general IS practitioner)	Q3 (personal experience)
Thinking	8	8
Understanding	7	8
Formulation	5	3
Modelling	1	0
Research	0	3
General knowledge	0	5

4 Summary and Future Research

To make logic and formal methods more central in the IS curriculum, further empirical evidence on the benefits of such courses for practitioners is required. While IS practitioners rarely apply formal methods directly, but rather use tools where they are "hidden" [5], exploring the effect of such courses on the cognitive processes of students seems the most fruitful direction. Our results highlights the potential of the methodology of exploring students' perceptions and attitudes in this context. This could be beneficiary for the education community also from another aspect. While the key role of abstract thinking for acquiring computation-related skills has been stressed by Kramer [6], ways of teaching it are still not well understood (some relevant general suggestions can be found in [7]). The positive effect formal methods courses may have on the cognitive processes of the students provides a good starting point for exploring concrete ways in which we abstract thinking can be taught.

We plan to further extend the data collection and analysis to larger student populations, including both undergraduate and graduate students taking relevant courses, as well as to experienced IS practitioners. In future research we plan to further investigate the impact of factors such as industrial experience and maturity of the subjects of the study. We also plan to reformulate the questions to allow a further sharper quantitative analysis.

The observations of this paper also bring forward the somewhat controversial notion of *computational thinking* (CT). This is a term introduced by J. Wing in 2006 [20], describing the mental activity in formulating a problem to admit a computational solution, which is crucial for many disciplines at our times. Naturally, it has also been pointed out as a key capability for future IS practitioners [3]. Note the striking relation of some of the above mentioned responses to key attributes of what J. Wing classifies as computational thinking [20]: (i) reformulating a seemingly difficult problem into one we know how to solve; (ii) using abstraction and decomposition when attacking a large complex task or designing a large complex system; and (iii) choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. The link between teaching formal methods and the development of computational thinking skills deserves further exploration (also due to the increasing public interest in CT).

References

1. Barland, I., Felleisen, M., Fisler, K., Kolaitis, P., Vardi, M.Y.: Integrating logic into the computer science curriculum. In: Annual Joint Conference on Integrating Technology into Computer Science Education (2000)
2. Glass, R.L.: A new answer to how important is mathematics to the software practitioner? IEEE Softw. **17**(6), 136 (2000)
3. Hardy, G.M., Everett, D.L.: Shaping the Future of Business Education: Relevance, Rigor, and Life Preparation. Palgrave Macmillan, London (2013)

4. Harvey, V.J., Wu, P.Y., Turchek, J.C., Longenecker, H.E.: Coordinated topic presentations for information systems core curriculum and discrete mathematics courses. In: Proceedings of ISECON 2005 (2005)
5. Hussmann, H.: Indirect use of formal methods in software engineering. In: ICSE-17 Workshop on Formal Methods Application in Software Engineering Practice, Seattle (WA), USA, pp. 126–133. Citeseer (1995)
6. Kramer, J.: Is abstraction the key to computing? Commun. ACM **50**(4), 36–42 (2007)
7. Kramer, J., Hazzan, O.: The role of abstraction in software engineering. In: Proceedings of the 28th International Conference on Software Engineering, pp. 1017–1018. ACM (2006)
8. Lethbridge, T.C.: What knowledge is important to a software professional? Computer **33**(5), 44–50 (2000)
9. Makowsky, J.A.: From Hilberts program to a logic tool box. Ann. Math. Artif. Intell. **53**(1–4), 225–250 (2008)
10. Page, R.L.: Software is discrete mathematics. ACM SIGPLAN Not. **38**, 79–86 (2003)
11. Patten, M.L.: Questionnaire Research: A Practical Guide. Pyrczak Publisher, Glendale (2001)
12. Skevoulis, S., Makarov, V.: Integrating formal methods tools into undergraduate computer science curriculum. In: 36th Annual on Frontiers in Education Conference, pp. 1–6. IEEE (2006)
13. Kelley Sobel, A.E.: Empirical results of a software engineering curriculum incorporating formal methods. ACM SIGCSE Bull. **32**(1), 157–161 (2000)
14. Kelley Sobel, A.E., Clarkson, M.R.: Formal methods application: an empirical tale of software development. IEEE Trans. Software Eng. **28**(3), 308–320 (2002)
15. Sotiriadou, A., Kefalas, P.: Teaching formal methods in computer science undergraduates. In: International Conference on Applied and Theoretical Mathematics (2000)
16. Tavolato, P., Vogt, F.: Integrating formal methods into computer science curricula at a university of applied sciences. In: TLA+ Workshop at the 18th International Symposium on Formal Methods, Paris, Frankreich (2012)
17. Topi, H., Valacich, J.S., Wright, R.T., Kaiser, K., Nunamaker, Jr., J.F., Sipior, J.C., de Vreede, G.J.: Is 2010: Curriculum guidelines for undergraduate degree programs in information systems. Commun. Assoc. Inf. Syst. **26**(1), 18 (2010)
18. Wing, J.M.: Teaching mathematics to software engineers. In: Alagar, V.S., Nivat, M. (eds.) AMAST 1995. LNCS, vol. 936, pp. 18–40. Springer, Heidelberg (1995)
19. Wing, J.M.: Weaving formal methods into the undergraduate computer science curriculum. In: Rus, T. (ed.) AMAST 2000. LNCS, vol. 1816, pp. 2–7. Springer, Heidelberg (2000)
20. Wing, J.M.: Computational thinking. Commun. ACM **49**(3), 33–35 (2006)
21. Zamansky, A., Farchi, E.: Teaching logic to information systems students: challenges and opportunities. In: Tools for Teaching Logic (2015)