Teaching Formal Methods in Computer Science Undergraduates

A. SOTIRIADOU and P. KEFALAS
Department of Computer Science
CITY Liberal Studies, Affiliated Institution of the University of Sheffield
13 Tsimiski Street, 546 24 Thessaloniki,
GREECE
{sotiriadou, kefalas}@city.academic.gr URL: http://www.city.academic.gr/

Abstract : - Formal Methods refer to a variety of mathematical modeling techniques, which are used both to model the behaviour of a computer system and to verify that the system satisfy design, safety and functional properties. The incorporation of a Formal Methods course in the undergraduate Computer Science curriculum is strongly suggested by scientific societies such as ACM, IEEE and BCS. In this paper, we record out experience of teaching the 2nd year Computer Science students Formal Specification, using the Z notation as a paradigm. We present a methodology for teaching formal specification as well as criteria for setting and assessing students work. We argue that students, through a rigorous mathematical approach to system specification, acquire knowledge, skills and abilities that are useful in all courses of the curriculum as well as in their professional lives as Computer Scientists.

Keywords: - Computer Science Curriculum, Teaching and Learning, Formal Methods, Formal Specification.

1 Introduction

Software reliability is a desirable property, especially in safety critical systems. To this end, the promotion of formal methods in the software development process is a promising approach [1,2]. Therefore, the introduction of formal methods into the education of the future computer scientists is vital for the success of formal methods [1, 3, 4, 5].

Scientific societies such as the Association for Computer Machinery (ACM), the Institute of Electronic and Electrical Engineering (IEEE-Computer Society) [6], and the British Computer Society (BCS) [7] list formal methods as one of the concepts that an undergraduate programme in Computer Science (CS) should incorporate.

In this paper, we record our experience on the twofold role that a Formal Methods course plays in the CS undergraduate curriculum. We believe that such a course is of extreme importance to an undergraduate CS student, and do not only contribute to the student's knowledge but most importantly to the appreciation of formal specification and also to the skills acquired through the corresponding learning process.

In the next sections, we present the goals of the specific course in formal methods. We then provide a justification for adapting the notation used, i.e. the Z specification language. A description of the teaching method and the teaching material follow. Finally, we discuss ways of assessing student performance and the opinions of the students for the particular course.

2 Aims of the course

Formal methods refer to the variety of mathematical modelling techniques that are used to specify and model the behaviour of a computer system and to verify that the system design and implementation satisfy system functional and safety properties. These specifications and verifications may be done using a variety of techniques and with various degrees of rigor.

Formal specification is the use of notations derived mostly from formal logic in order to describe assumptions about the world that a system will model, requirements that the system should achieve and a design to achieve those requirements.

Many CS departments of traditional Universities have incorporated formal methods in their curricula. Most of those courses cover aspects of both formal specification and formal verification, following a breadth-first approach. Our approach, however, in designing the specific formal methods course incorporates solely the formal specification component, which is taught in depth. There are a number of reasons for suggesting the latter. First, for many CS students it is already difficult to conceive that mathematics and mathematical modelling play an essential role in Computer Science. Secondly, during their first year of studies students have completed among others, courses in Structural and Object Oriented Programming, Programming Methodology and Systems Analysis and Design. Our experience shows that by the beginning of the second year of their studies they

have not appreciated yet the importance of specification, yet alone of formal specification in software lifecycle. Of course, there are ways to prepare the students in order to receive their first introduction to formal methods, mainly by the Discrete Mathematics course, which are discussed elsewhere [8]. However, we believe that, even with an extensive exposure to Logic and Set Theory, they have not developed yet the necessary intellectual abilities in order to be exposed in many different issues on formal methods. Lastly, our approach coincides with other proposals [4] which state that it is preferable to emphasize depth in early courses in formal methods.

The major aims of the course we offer is to assist students:

- Abandon their attitude that equates the development of a system with the task of writing code; make the students think very carefully about the unambiguous description of the system and appreciate the significance of formal specification in the development of reliable systems.
- Understand that in the process of developing reliable systems, formal specifications offer the opportunity to understand better the system under development, by revealing inconsistencies and ambiguities that could otherwise remain undetected.
- Recognize that the exposure of those inconsistencies is important in early stages of the development of a system.
- Realize that formal specifications contribute in enhancing the communication between client and developer, when insights obtained from the mathematical model initiate further discussions with the client.
- Learn how to design a complete and well-structured mathematical model of a system in a widely used notation, namely the formal specification language Z.
- In the acquisition of skills important in CS education. A crucial advantage of formal specification is that it permits for abstraction, that is, the description of systems without dealing with implementation details. Therefore, in a formal methods course students are exposed to the process of abstraction, which according to [6] should be a prominent concept in the CS undergraduate curricula.

3 Choosing the Notation

Our philosophy for designing such a course is not to simply teach a specific formal notation but rather to address the issues of mathematical modelling and those important processes, such as abstraction and problem solving. Nevertheless, a vehicle with which these concepts are taught should be chosen. We have adopted a notation, which we believe best reflects the aims of the course. The chosen notation is the Z specification language.

Z is a formal specification language, which has been developed at the Programming Research Group at the Oxford University Computing Laboratory since the late 1970s. It is used by industry as part of the software (and hardware) development process. Z has been successfully used to specify many software systems including database systems, transaction systems, distributed computing systems, and operating systems [9].

Z is a formal specification notation based on set theory and first order predicate logic. Together they make up a mathematical language, which is only one aspect of Z. For a more detailed description of Z, the reader may consult [10,11].

The following characteristics of Z are the main reasons that justify our decision to choose it as the notation for the course:

- Z is a very popular language both in the Industry and in the Academia [3].
- Our students have already studied during their first year the mathematical concepts of logic and set theory, needed to construct specifications in Z.
- One of the advantages of Z, is that allows for both *procedural abstraction*- separation of *what* a program does from the details of *how* it does it- and *data abstraction* - abstracts details of the representation of data away from any particular programming language data structures. Therefore, students are explicitly exposed to and utilize the abstraction process.
- Another aspect of Z is the way in which the mathematics can be structured. *Schemas* form the backbone of Z specifications, helping to structure and modularise specifications; the schema language can be used to describe the state of a system, and the ways in which that state may change. In Z, schemas can be referred to and composed with other schemas, hence a schema is analogous to the idea of a module or subprogram in a programming language.
- In addition, this graphical notation of Z, schemas, aids the development of readable and

well-structured specifications. This is important to students especially when they read specifications developed by others.

- The schemas are used to describe system properties. Therefore, students are required to look for and describe essential properties of a system such as invariants and pre-conditions.
- A characteristic feature of Z is the use of types. Every object in the mathematical language has a unique type, which provides a useful link to programming practice. All types in Z can be considered as sets. They consist either of *basic type*, this is either predefined or user-defined set, or a *complex type* made up from the basic types.

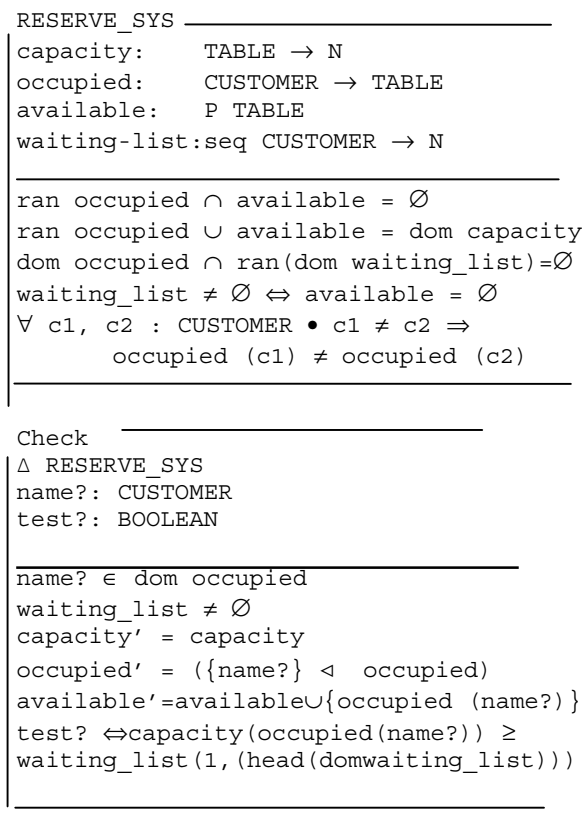

Fig. 1: System State Schema and the Operation's Check Schema of the Specification of a Restaurant's Reservation System.

The general format of a schema is a box, which includes two parts separated by a line segment: the *declarative part* and the *predicate part*. Fig. 1 presents a Z schema that might be part of a specification for a restaurant's reservation system responsible for monitoring the assignment of customers to tables. We may introduce the basic types `[TABLE, CUSTOMER]` to define the "universal" sets of all possible tables and customers.

The declaration part of a schema includes the data abstraction. The declarative part of the reservation system declares a set and three functions, where the `waiting-list` has a sequence as its domain and represents the waiting list of customers together with the size of their corresponding party. The predicate part of a schema includes predicates that need to be true if the requirements of the system are to be satisfied. The first predicate in the predicate part of the security system ensures that a table should be either available or occupied by a customer, not both.

In addition, in Fig.1 the specification of the `Check` operation is presented, where given the name of a customer who has completed his dinner, the given customer and his party leave, and the first customer in the waiting list is checked whether could sit in the freed table or not; that is, it is examined whether the size of the customer's group could fit in the freed table.

4. The Teaching and Assessment Method

In our undergraduate curriculum, we have included the Formal methods course in the 2nd year of studies, taught within one semester, i.e. approximately 36 conduct hours, including lectures and tutorials.

4.1 The Teaching Method

In order to fulfill the aims of the course and to help students enhance their reasoning and problem solving skills our teaching approach includes the following:

- The number of students in a classroom does not exceed 25 and during tutorials two lecturers are present, keeping in that way a rather low student-faculty ratio. This enables the creation of a classroom environment that supports active learning [5]. The active learning approach is adopted both during lecture hours and tutorials.
- During a regular two hours lecture initially new topics are introduced and then how they utilized through a specific case study. However, the specification of the case study is not produced by the lecturer. It is rather the output of an ongoing interaction with students. The role of the lecturer is to pose key questions and to guide students in producing the formal specification. In this way students are actively involved in the process and are not passive receivers of information. Even when a

student's approach is not appropriate it is left to the rest of the students to uncover the deficiencies.

- During classes students form 2-3 members groups and they work on specific case studies assigned to them. The cases are presented to students in natural language and include ambiguities intentionally. As students proceed in constructing the formal specifications those ambiguities are revealed. Then the students need to discuss those ambiguities with the lecturers, who play the role of clients. Therefore, students realize and appreciate the contribution of a formal specification in the clarification of the systems requirements and in aiding the software development process.
- In some tutorials students are presented with Z formal specifications of systems and are requested to understand them and interpret them in natural language. In this way students enhance the skill that a software developer need to not only produce but also to read and understand a specification.
- In addition, students are presented form early lectures with a specific methodology on how to approach a construction of a formal specification in Z. This includes issues ranging from how to read the specification in natural language and define the basic types, up to steps that should be followed in constructing the system state schema and operations schemas for schema calculus. We believe that this explicit guidance is important instead of leaving students extract the methodology from the lectures on their own.
- It is sometimes useful to begin with semiformal specifications in order to relief the students to learn the notation at early stages and focus on specification issues. It is an extra burden for students to familiarize with the syntax of Z, and therefore the approach of semiformally specifying a simple problem is adopted.

4.2 Content and Reading Material

During the first couple of weeks properties of good quality software, impediments for developing reliable systems, and reasons that led to the software crisis and the emergence of Formal Methods are discussed, followed by a brief review of logic and set theory. We proceed with the Z notation, schemas and schema operations. During the remainder of the course we consider specifications using sets, functions, bugs, sequences and relations.

In the past we were suggesting specific textbooks for our students to buy and additionally we were providing them with lecture notes [12]. For a number of years, students' evaluations of the course revealed that they based their study solely on our notes. Therefore, we do not suggest any more a specific textbook but rather we provide a reading list, which consist of a number of books and articles. In addition, we have developed a web page for the specific formal methods course, from where students can acquire, the lecture notes, the case studies we discuss during lectures and tutorials, links to other interesting formal methods sites, and a frequently asked question page.

4.3 Assessment

Assessment of students consists of two parts: coursework and examination, contributing to their final grade by 30% and 70% respectively.

The coursework is divided into two assignments with different objectives:

- the first is focused on reviewing set theory and predicate logic and on the development of the semiformal specifications of two information systems.
- the second is focused on the development of formal specifications in the Z language of simple systems.

The students know the criteria for assessment in advance. This is useful in order for them to concentrate on specific issues during their work of specifying a system and not bother with unnecessary details, which are not relevant to the course aims. The criteria set in the assignment handout requires from the students to:

- understand a given formal specification of a system state and being able based on this specification to produce formal specifications of operations for that system.
- understand the possible constraints of a system and interpret them using formal language into the invariants of the system.
- design complete and well structured Z specifications of systems.
- produce appropriate data and procedural abstractions.
- construct robust formal specifications.
- describe and explain the rational of using specific concepts for the Z specification they propose.

On the other hand, the examination paper is set in such a way that a student should demonstrate both

understanding and ability to construct a formal specification. This is achieved through the division of the number of exercises to those that the students should explain and justify a given specification and those in which the students, given an initial informal description of a system, are requested to write Z notation for specific operations. Our experience in assessing the exam paper showed us that it is rather difficult to assess an answer, which includes a specification written from scratch. It is, therefore, advised to potential lecturers of the course to give away to the students at least part of the system state schema on which their specifications will be based. Otherwise, the possibilities of correct specifications are endless, with the students temped to construct the most complicated ones. Their choice on system state is rather crucial for the time needed to answer an exam question while their answer may not satisfy the question's objectives.

5. Evaluation and Conclusions

We have presented a way to incorporate Formal Methods in the Computer Science Curriculum as well as a methodology for teaching and assessing such a course.

The validity of our approach is demonstrated by two factors: (a) the positive comments made by the external examiners through a series of years, and (b) the feedback from students. The latter is acquired by the student evaluation questionnaires, which are distributed to the students at the end of the semester. A comparative study showed that the students believe that the Formal Methods course has increased their knowledge and skills (3.9 and 3.8 in a scale of 5) more than the rest of the course run in parallel (Java, Logic Programming, and Operating Systems). In addition, students by the end of the course think that formal specification has rather less difficulty than the concepts introduced in other courses. This result is encouraging, since the students are often prejudged against any mathematical course.

Concluding, we would like to urge our colleagues of other Universities to adopt Formal Methods as a means to convince students that Computer Science is not just about Programming, and also as a vehicle to introduce concepts and exercise skills, which are spread out not only in all Computer Science courses but also in the future professional careers.

References:

[1] E. C. Clarke, J. M. Wing, Formal Methods: State of the Art and Future Directions, *ACM Computing Surveys*, Vol. 28, Number 4es, pp. 116-116, December 1996.

[2] A. Harry, Formal Methods Fact File: VDM and Z, Wiley, 1996

[3] C. J. Burgess, The Role of Formal Methods in Software Engineering, Education and Industry, Proceedings of the 4th Software Quality Conference, pp. 98-105, July 1995.

[4] Report on the 21st Century Engineering Consortium Workshop ©, A forum on Formal Methods Education, Melbourne Florida ,1998.

[5] H. Walker, H. C. Cunningham, and R. Davis, D. Troeger, Formal Methods in the Undergraduate Computer Science Curriculum, Proceedings of the 26th Technical Symposium on Computer Science Education, SIGCSE Bulletin, Vol. 27, pp. 398-399, ACM Press, March 1995

[6] A. Tucker (Ed.), Computing Curricula 1991: Report of the ACM/IEEE-CS Joint Curriculum Task Force, 1991.

[7] *Course Accreditation*, The British Computer Society, June 1996

[8] A. Sotiriadou, P.Kefalas, Logic and Sets in the Computer Science Curriculum, Proceedings of the 2nd Panhellenic Logic Symposium, Delphi, pp. 191-196, July 1999

[9] I. J. Hayes (ed.), Specification Case Studies. Second Edition. London: Prentice-Hall, 1992.

[10] J. Bowen, [J.P.Bowen@reading.ac.uk], *Information about the Z formal specification notation*, [http://www.cis.ohio-state.edu/ hypertext/faq/usenet/z-faq/faq.html]

[11] J. M. Spivey, The Z notation: a reference manual, Prentice Hall International, 2nd Edition, 1992.

[12] A. Sotiriadou, Formal Methods Lecture Notes, [http://www.city.academic.gr/intranet/ courses/BSc_Computer_Science/csd2300/ Pages/FMNotes.htm]