Teaching the Oxford Brookes Formal Specification Module

Ian Bayley, David Lightfoot, Clare Martin
School of Technology
Oxford Brookes University, United Kingdom
{ibayley, dlightfoot, cemartin}@brookes.ac.uk

Abstract

We teach a second-stage undergraduate module called *Formal Specification* using the Z notation to students who do not typically have a high level of mathematics. We have done this with a good degree of success for many years and the module continues to be viable. We discuss the pattern of our course and speculate on why it is successful.

1. HISTORY

The teaching of formal specification at Oxford Brookes University began as part of an HND/HNC¹ in Software Engineering at the then Oxford Polytechnic. The material from that course was developed into the second-stage undergraduate module called Formal Specification. We have taught this module every year for many years and have a good degree of success measured as marks scored, pass-rates and student feedback. To date we have taught this module to over 600 students.

2. PLACE OF THE MODULE IN THE COMPUTING COURSE

The module is only compulsory for our specialised Computing Science field, which is taken by a small number of students. It is also available to all students on computing courses and is taken up by a sizeable minority. Most of the students who take this module have chosen to do so.

3. CHOICE OF NOTATION

We chose to cover only state-based specification in the limited time available to this module. We chose the Z notation because of its relative simplicity (when compared to VDM). When the module began only a small number of books on Z was available, so the original course leader wrote a simple, inexpensive textbook, *Formal Specification Using Z* [1]. This has been the principal teaching text throughout the course's life and it is used on similar courses at some other institutions.

At various times in recent years we have considered changing the notation to AMN (B) [2]. This would offer the advantages of the notation's being machine readable and of animation of specifications. However, although we use them successfully with small classes of MSc students, the available software tools are not really intended as educational and we have so far feared that the advantages would be outweighed by the difficulties we expect undergraduate students might encounter in using the tools.

4. BOOKS RECOMMENDED

In addition to *Formal Specification Using Z* we direct students to Mike Spivey's Z reference manual [3] for detailed and thorough coverage of the notation. This is available as a reference in the university library and also, since out of print, from the Internet.

As an alternative introductory text we suggest Sinclair, Potter and Till's introduction to Z [4].

5. PREREQUISITE KNOWLEDGE

A successful result in a first-stage course *Discrete Mathematics* (or its equivalent from another institution) is a prerequisite of study of the formal specification module. This module on discrete

---

¹ Higher National Diploma / Higher National Certificate: British educational qualifications
Teaching Formal Methods: Practice and Experience, 15 December 2006

mathematics is very ably taught by the Department of Mathematics. Nonetheless some time will have passed since the student's study of that module and we find it judicious to revise briefly those elements relevant to formal specification. In the practical classes we provide many examples of expression evaluation to exercise and reinforce understanding of notation.

6. COURSE STRUCTURE

From the start the course was organised with two key principles: emphasising the 'constructive' nature of Z and interleaving theoretical and practical material.

6.1. Constructive nature of Z

The course structure reflects the constructive nature of the mathematics behind the Z notation. For example, when teaching relations we emphasise that a relation can be considered as a set of pairs and so all the ideas learned about sets are applicable. This is emphasised each time a new Z construct is introduced and can be summarised as: a sequence is a function, is a relation, is a set of pairs. This explains, for example, why in Z one can write $\#s$ to find the *cardinality* of a set s , or the *length* of a sequence s . We find that emphasising this encourages students because it shows that a new topic is just a small extension of what has been learnt already.

6.2. Interleaving of theory and practical

We were concerned that a student without much confidence in their mathematical ability might become disheartened if theoretical material was presented without some 'leavening'. We interleave theory and practical by showing how the theoretical topic just introduced can be applied in a simple, everyday scenario.

For example, when teaching relations we show a simple schema containing just one or two relations and illustrate them with simple diagrams. We also make extensive use 'potato diagrams' because we find that students like and understand that style.

6.3. Teaching pattern

In common with other modules taught in the department each module consists of 11 two-hour lectures and 11 one-hour practical classes.

We cover most of the Z notation very quickly, though with a pause for breath before taking on the considerable amount of notation concerning relations. The aim of covering it quickly is to allow the students time to *apply* the notation both in case studies we present to them and in their own exercises and coursework.

A typical schedule of topics is as follows:

Week	Topic
1	Sets, State schemas
2	Propositional logic, schema operations
3	Predicate logic
4	Z/EVES tool, consolidation exercises including past exam questions
5	Relations
6	Functions
7	Sequences
8	Case study Practical: coursework
9	Extending a specification, schema calculus, error handling
10	Case studies
11	Past exam papers and revision

6.4. Topics not covered

In a short course such as this it is necessary to be selective. We have chosen to omit the following:

- Bags
- Binding operator θ (though we explain and use $\exists$ schemas)
- Promotion
- Tuple selection
- Free types

- *if then else* expressions (we prefer pure propositional calculus)
- *let* expressions
- λ and μ expressions
- Proof
- Generic schemas

7. CASE STUDIES

We have developed a series of case studies for teaching purposes. We try to choose systems to specify that will be related to other areas of a student's work, such as formally specifying a programming coursework that they have undertaken. We also try to produce examples set in an everyday context. Some of case studies we have produced are:

- The (*inevitable*) Library Specification from [1]
- The (*not-so-inevitable*) Comparison of Library Specifications
- The Aircraft Specification from [1]
- The Text Editor from [1]
- Eurovision ("*Voici les votes!*" – *Formal Specification as light entertainment* [5])

8. SOFTWARE TOOLS

In the early years of the module we used no software tools. Later we made available a Z font and continue to do so. Since encountering Z/EVES at the Teaching Formal Methods workshop in 2003 we have used the tool extensively and have made it available to students. It offers advantages such as typesetting, syntax checking, type checking and checking that schema inputs are within the domain of the functions to which they are applied. Unfortunately this tool is no longer supported. We are considering adopting CADiZ.

9. ADVANTAGES OF NOT USING SOFTWARE TOOLS

We do not *require* the students to use software tools. One reason is that we perceive an advantage to delaying their use: When a student makes an inaccurate attempt at an exercise a software tool is likely to produce a fairly cryptic and discouraging response. The reverse is also true in that there is a strong 'reflex action' for the student to assume that the Z they have written is correct just because the tool says that it is *syntactically* correct. In a class a sympathetic lecturer or practical assistant can apply intelligence to understand what the student intended and correct the attempt and explain the error without demoralising the student. We think that this might account for the frequently heard opinion from students that they like formal specification but do not like programming.

10. ASSESSMENT

10.1. Coursework

We ask the students to work in small groups to produce a formal specification of a case study of their own choice. We offer guidance on this choice to make sure that the students attempt something appropriate and within their abilities. Making it the students' choice has the advantages of increasing student motivation, individualising the assignment, encouraging collaboration and eliminating plagiarism. For the sake of fairness, a proportion of the marks is related to the difficulty of the system attempted. We encourage the students to seek help from the lecturers on the problems they encounter. This all dramatically cuts the preparation time for assignments without having any significant impact on the marking time. We have also tried, without success, to encourage the students to specify real-life working software that already exists, such as that of common websites. We think this might be because in some example specifications the data items are easily identifiable as business-domain objects (such as shop assistants and stock items) instead of being more abstract items hidden inside a database behind a server.

10.2. Examination

The format of an examination paper (two-hours) requires the student to answer any three questions out of four. Questions typically start with a comprehension section in which students

are required to explain what is meant by some fragments of a Z specification. This is followed by their being required to specify some further operations. We almost always give the state schema. This has advantages of presenting the student with a correct basis for subsequent operations they will be asked to specify and also means that all student answers will use the same identifiers.

11. STUDENT PERCEPTION OF THE MODULE

Students typically report satisfaction with the module. They find it interesting and well structured. Many who do not enjoy computer programming seem to enjoy formal specification. Some find the pace of the early weeks too fast. Some say that is 'does not need too much background knowledge' and that 'the course covers everything (*that they need to know*)'.

12. SUMMARY OF RESULTS OVER THE PAST 10 YEARS

Year	Number of students	Average mark %	Pass rate %
2007	(registered to date 29)	—	—
2006	29	59	79
2005	42	66	81
2004	82	63	86
2003	61	54	80
2002	65	55	77
2001	101	48	60
2000	85	60	81
1999	91	51	70
1998	53	53	73
1997	8	48	100

The average mark is typically high, as is the pass rate. The results of module runs in previous years are made available to students and we suspect that some of them choose their modules hoping that the good average mark and pass rate in earlier runs of a module will mean that they will necessarily do well. This might explain the high take-up of the module in 2001, which unfortunately had a disappointing pass rate.

13. OTHER MODULES USING FORMAL METHODS

We have also successfully introduced the ideas of specification by pre- and post-conditions to the descriptions of examples and assignments of our programming modules. We have also successfully introduced the use of assertions to those modules.

Our MSc courses include a challenging but popular module called *Computer-Aided Software Development* that teaches specification and development of software by the B Method and which makes use of B-Core's B Toolkit.

An MSc module called Software Engineering for students without a first degree in computing includes an introduction to formal specification. This module is also well received.

We also teach elements of the module in the *English Week* of IUT-2 Grenoble, France and as part of a course *Formal Specification and Development of Software* supported by the European Union and delivered at Johannes Kepler University, Linz, Austria.

14. PLANS FOR THE FUTURE

We are convinced that the ideas introduced by a study of formal specification have a profound positive effect on students' modes of thinking about software specification and development. We continue to try to spread the use of concepts from formal methods to other modules in our computing courses by trying to provide convincing examples of their utility. For example, in programming modules, for assignments where students are asked to implement procedures, specifications are given using pre- and post-conditions. To reinforce best practice, example procedures given in lectures are documented in this way.

We would like to increase our use of suitable software tools and will periodically reconsider the best notation to use.

ACKNOWLEDGEMENTS

The course is currently taught by the authors of this paper. Over the years the lecturing team has included former members of the department Jeremy Gibbons and Joy Reed. Practical classes have also been ably conducted by the lecturers and by many research students, including Zaineb ben Fredj, Jérôme Mengin and Christine Poerschke.

We are also grateful to the reviewers of the first draft of this paper for their helpful suggestions.

REFERENCES

- [1] Formal Specification Using Z, David Lightfoot, Palgrave 2001
- [2] The B-Method: An Introduction, Steve Schneider, Palgrave, 2001
- [3] The Z Notation: A Reference Manual (2nd Edition), J M Spivey, Prentice Hall, 1992
- [4] An Introduction to Formal Specification and Z (2nd Ed), B Potter, J Sinclair and D Till, Prentice Hall 1996
- [5] Teaching Formal Methods 2003, <http://cmsdomino.brookes.ac.uk/tfm2003/>