

Specification: much more than learning a language

S. Green, J. Webb

*Department of Computing, University of the
West of England, Bristol, UK*

Abstract

During the last academic year a module introducing formal specification was presented for the first time at the University of the West of England. The course, based upon Z [1], was delivered over a 12 week period to 150 first year HND and degree students using one lecture, one tutorial and one laboratory session per week.

A number of problems were encountered during this first presentation of the module. The main problem was that students found it difficult to create a complete specification from a set of requirements written in English; many did not know either how to get started or how to proceed once they had started.

To help the students to develop specifications it is proposed that in future years they should be taught a method of analysing a statement of requirements that will produce a good starting point from which to begin to create specifications in Z. In addition they should also be taught a method that will guide them through the process of completing a specification in Z.

1 Introduction

At UWE a small group of lecturers from the faculty of Computer Studies and Mathematics has recognised the importance of formal methods in software engineering, and is now in the process of creating and delivering a coherent set of modules—"Discrete Mathematics", "Introduction to Formal Specification", "Discrete Mathematics for Formal Systems", and "Advanced Formal Specification"—to teach formal methods.

During the last academic year both the "Discrete Mathematics" and the "Introduction to Formal Specification" modules were presented for the first time. The latter module, based upon Z [1], was delivered over a 12 week period to approximately 150 first year HND and degree students using one lecture, one tutorial and one laboratory session per week. Lectures presented topics ranging from data types through to an introduction to data refinement and operation decomposition; tutorials reinforced the content of lectures with set exercises; and laboratory sessions were used to teach the students how to use the CADiZ tool to create and type-check Z specification documents (CADiZ is a Trademark of York Software Engineering Ltd., York).

A number of problems were encountered, the main one being that students found it difficult to create a complete specification in Z from a set of requirements written in English. This problem has been analysed and some suggestions have been made for tackling it.

The paper begins by describing both the "Introduction to Formal Specification" module—its aims, prerequisites, structure, content, delivery, and assessment—and the students who studied it. Next, the main problem associated with its delivery, i.e. the students' inability to first begin and then complete a Z specification, is presented. Finally, some ideas about how to alleviate this problem are suggested.

2 The module

2.1 Aims

We hoped that by the end of the module the students would appreciate the need for, and the rôle of, formal methods in software engineering; be aware that each type of specification language is suitable for specifying a particular type of problem; be able to use the Z specification language to create specifications from requirements expressed in English; know how Z specifications are transformed into equivalent program code; and know why proof is important in formal methods, and have some feel for how proofs are carried out.

2.2 Prerequisites

All students taking the “Introduction to Formal Specification” module in the second semester had already completed the prerequisite “Discrete Mathematics” module in the first semester. Thus, before starting the former module, the students already had a basic knowledge of mathematical modelling, sets, relations, functions, sequences, propositional logic, predicate logic, and graphs.

2.3 Structure, content, and delivery

We chose Lightfoot’s book [2] as the main course textbook as it was felt to be easy to read and understand, and therefore likely to be used by most of the students. The content of the lectures was based upon the chapter structure of Lightfoot’s book, but with some modifications to include additional material such as data types, data refinement, and operation decomposition. The order and content of the lectures was as follows: “The context for formal specification”, “Types, sets, and modelling”, “Proofs in propositional logic”, “Schemas”, “Proofs in predicate logic”, “Relations, functions, and sequences (parts 1 and 2)”, “Data refinement”, “Operation decomposition”, and “Misconceptions surrounding formal methods”.

Each lecture presented relevant information and posed questions designed to make the students think about the lecture topic. Students attempted to answer the questions during a lecture, with the correct answers eventually being supplied by the lecturer. The information and questions were presented on overhead-projection slides. Each student received a copy of the slides at the start of each lecture.

The content of each lecture was reinforced by complementary tutorials attended by groups of up to sixteen students in which prepared exercises were attempted. A tutor was available in each tutorial to help each student individually, and also to discuss common problems with the whole group. At the end of each tutorial a copy of the correct answers (plus explanatory notes) was issued to each student.

A series of weekly laboratory sessions was intended to guide the students gently into using the CADiZ tool for formatting and type-checking Z specifications. The series began with an introduction to Unix and proceeded by way of Emacs, Latex, Z in Latex, and finally CADiZ.

2.4 Assessment

The students were assessed both by an assignment during the module and by a formal examination at the end. For the assignment, the students were

426 Software Engineering in Higher Education

presented with a simplified version of the classical library database problem [3] and asked to create a Z specification for it. To help with the assignment, students were directed both to chapter seven of Lightfoot [2], which outlines the process of creating a relatively complete specification, and to the third appendix at the end of the book, which contains hints on how to create specifications. The students also received notes produced by the module team advising them first to familiarise themselves thoroughly with the requirements outlined in the problem, then to develop an appropriate state schema, next to develop operation schemas, and finally to develop associated error schemas.

3 Students

The module was taken by both Higher National Diploma (HND) and honours degree students. The former group are studying for an HND in Computing; the latter group are studying either for a BSc in Computing or a BSc in Computing for Real-Time Systems.

4 Problems

Several problems with the outcome of the first delivery of the module were perceived by the module team. These included poor examination performances by students studying for either the HND or BSc in Computing, and very little time spent by the students in attempting to master the specification support tool, CADiZ.

The main problem, however, was that students felt themselves unable to tackle the module assignment (described above). After questioning the students it became clear that they did not have a clear idea of how to begin the task of producing a specification from a set of requirements expressed in English, nor did they have a clear idea of how to proceed with the task once they had started.

5 Solutions

5.1 The need for prior knowledge

During the course of delivering the formal specification module, it became

obvious that students would benefit from knowledge of the aims and practices of problem analysis. Although students had studied both analysis and data modelling in other modules, which ran in parallel with the formal specification course, it was felt that a useful appreciation of the importance of analysis was not achieved until too late in the formal specification course.

One of the authors had previously observed that students with prior knowledge of the aims and techniques of analysis adapt well when confronted with formal methods. This was thought to be due to their ability to apply this existing knowledge to the novel formal methods concepts. Such students usually have had between one and two years' experience in practising analysis skills.

This leads to the suggestion that at UWE it might be wise to delay the introduction of formal specification until the second year, thus allowing students a longer period to digest the aims and techniques of conventional analysis presented in the first year.

Alternatively, it is possible that an analysis and data modelling module could be scheduled in the first semester of the first year, as a prerequisite to the formal specification module being presented in the second semester.

5.2 Module content strategies

Even assuming a prior knowledge of analysis techniques, there remains the problem of how best to help the students to exploit it when they are learning how to create formal specifications.

It has been noted on another, more advanced, course that even very good students will often not recognise the interconnections between each of the various diagrammatic and formal analysis methods as each method tends to be presented in an isolated way. We have found that formal specification students make much faster progress when they are explicitly taught the links between each method, and how to convert between the various diagrammatic methods and the formal specification language being studied, and vice versa. For example, relationships between entities, datastores and sets can provide useful insights into how a specification may be informally checked. The details of such techniques have been published (for example [4, 5]) but are perhaps not taught widely enough. Such techniques also provide a process by which students can feel their way into a problem by attacking it on several sides. A summary of these relationships is included later. Students should normally be taught the most useful relationships at a very early stage of the course.

Our current preference, based upon experience gained from teaching other

428 Software Engineering in Higher Education

modules, is for a method of analysis and specification production based on a combination of Entity Relationship (ER) diagrams, Venn Diagrams, Dataflow Diagrams (DFD) and Entity Life History (ELH) diagrams. All these diagrammatic methods are currently widely taught at UWE and students are expected to attain proficiency in their use.

An alternative strategy, described by Polack [6], might be to teach a combination of structured and formal techniques such as SAZ[6], which combines SSADM version 4 and Z. This could improve the linkage between diagrammatic and formal techniques. However, as this is a relatively new method, there might be drawbacks in terms of the level of staff training required and in overall student acceptance.

5.3 Mappings between diagrammatic and Z constructs

Some of the significant mappings that exist between our preferred diagrammatic constructs and Z constructs are presented below. Other mappings can be introduced to students as the need arises.

- ER diagram entities map to Venn Diagram supersets and thus to Z types
- ER relationships map to relations or functions within the Venn Diagrams and Z state schema
- DFD datastores map to Z sets, relations or functions
- DFD processes map to Z operation schemas
- Flows between DFD processes and external entities map to Z operation schema input and output variables

5.4 The proposed method

Based on the mappings presented above, an outline of the method we propose to teach to students is presented below. This method has the advantage of drawing upon students' prior learning, while also providing a useful reinforcement of the mappings listed above.

1. Produce an ER diagram from the prose requirement. The diagram will comprise entities and the relations between them. No attempt should be made to resolve any many-to-many or one-to-one relations that may arise.
2. Convert the ER diagram into a network of related Venn Diagrams. Do this by turning the ER entities into supersets; by deriving from the requirements sets within each superset [5]; and by converting the relationships into relations between sets. At this stage details of subsets

implied by the requirements may be entered onto the diagrams.

3. Convert the Venn Diagram into basic type declarations and a preliminary Z state schema. Do this by converting the Venn Diagram supersets to Z basic types; declaring the sets and subsets; and declaring Z relations or functions using the information given by the ER and Venn Diagram relationships. After this step a relatively complete state schema should result.
4. Produce a DFD from the English requirement. This diagram comprises external entities, processes, datastores and dataflows. During this step cross-checks are possible with both the ER and Venn diagrams, and with the state schema.
5. Derive operation schemas from the DFD processes. Each process will have dataflows into and out of it. Where these flows are from, or to, external entities, the flows convert into Z input, or output, variables [4] respectively. Select the Delta state or Xi state of each operation schema according to the direction of flows within the system boundary. Note that the datastores equate to the already defined sets or relations. After this step the state schema will be fully-defined and the operation schema declarations will be complete.
6. Produce an ELH for each distinct ER entity instance. ELH diagrams show the events and order of events happening to each entity instance.
7. Derive error operations and outline operation predicates from the ELH diagrams.

6 Conclusion

With the benefit of hindsight it is clear that our initial module focuses on teaching students the main aspects of Z as a language—schemas, types, operations, comments, etc.—but does not focus on teaching students how to use Z to create complete specifications from real-world requirements. As a consequence it was almost inevitable that our students would find it difficult to tackle an example of such a task.

We now recognise that our module should emphasise both Z as a language, and how to use Z to create complete specifications. Experience on other formal specification courses suggests that the acquisition of specification skills may be facilitated if the students have already been exposed to methods for analysing system requirements. We feel that the benefit of such exposure

430 Software Engineering in Higher Education

might be enhanced if they also have available a specification method which helps them to create the various parts of a complete Z specification from the results of analysing requirements.

In this paper we have identified some analysis diagram types (produced by well known analysis techniques) which we have already found to have been useful on other formal methods courses. In addition we have proposed the outline of a method which uses these diagram types to help to create a complete Z specification.

For the future, it is our intention to suggest that an analysis module should precede our module rather than run in parallel with it, and to further elaborate the proposed method and introduce it into our module.

References

- [1] Spivey, J.M. *The Z Notation: A Reference Manual*, Prentice Hall International Series in Computer Science, 1992.
- [2] Lightfoot, D. *Formal Specification Using Z*, Macmillan Press, 1991.
- [3] Babb, R. G. & Kieburtz, R. & Orr, K. & Martin, N. & Mili, A. & Gearhart, S. Workshop on models and languages for software specification and design, *Computer*, 1985, **March**, 103-108.
- [4] Randell, G. Dataflow diagrams and Z, in Z User Workshop 1990 (ed J.E. Nicholls), pp 216 to 227, *Proceedings of the 5th Annual Z User Meeting*, Oxford, England, 1990, Singer-Verlag, 1991.
- [5] Semmens, L. & Allen, P. Using Yourdon and Z: an approach to formal specification, in Z User Workshop 1990 (ed J.E. Nicholls), pp 228 to 253, *Proceedings of the 5th Annual Z User Meeting*, Oxford, England, 1990, Singer-Verlag, 1991.
- [6] Polack, F. Teaching the SAZ method: SSADM version 4 and Z, in Z Users' Meeting 1994 Additional Session (ed N. Dean), p 43, *Educational Issues of Formal Methods*, St John's College, Cambridge, England, 1994.