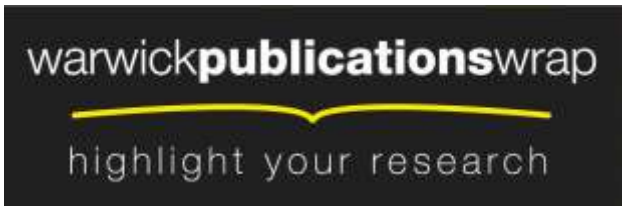

Experiences of Teaching a Lightweight Formal Method

R.C. Boyatt[1] and J.E. Sinclair[2]

Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK

Abstract

This paper reports our experience of using a "lightweight" formal approach, Alloy, and its associated tool support for teaching a core undergraduate module introducing formal methods. It considers the benefits and drawbacks in terms of both the student experience and our own aims and objectives for the module. In addition, we link the practical, experimental approach supported by the Alloy Analyzer to educational theory and consider the implications of such an approach to teaching and learning.

Keywords: Formal methods, Alloy, lightweight

1 Introduction

Despite an increasing focus on formal methods education in recent years, many institutions struggle to find a place for formal methods in the Computing (and even Computer Science) curriculum. In many cases, formal methods appear to be a decreasing component rather than progressing towards pervading the curriculum as advocated by many to be the ideal situation [Win00]. Reasons cited include the difficulty of convincing students and staff of the relevance and usefulness of formal methods; the "mathematics phobia" issue (together with falling entrance requirements and declining application numbers); pressure on the curriculum due to the range of required components and the desire to provide popular modules for our students (or customers). On the positive side, there are also reports of tools and teaching approaches finding acceptance in the core curriculum by their own merits (for example, Backhouse's "Algorithmic Problem Solving" approach [Bac06]). However, prevailing conditions remain unhelpful (for instance, in terms of student

[1] Email: rboyatt@dcs.warwick.ac.uk
[2] Email: jane@dcs.warwick.ac.uk

numbers [UCA07,Tod05]) and formal methods are likely to be under continued pressure.

At the University of Warwick, recent changes to the Computer Science degree have necessitated a rethink of our approach to teaching formal methods. In this paper we describe the changes made and consider the implications in terms of the student experience and what we can hope to achieve educationally. In particular, we examine our decision to investigate the use of a "lightweight" formal method and consider whether such an approach can provide an appropriate introduction. We identify the distinctive features of lightweight formal methods that are well-suited to a learning environment, and we identify areas in which caution may be needed. Our motivation was to reflect on and assess the impact of specific changes at one institution. This has led us to consider more general aspects of formal methods education, such as the way students interact with tools and the degree to which ideas in formal methods education align with certain strands of education theory.

2 The context for change

Until recently, Computer Science undergraduates at the University of Warwick studied a core Formal Methods module in their second year. Using the Z notation [Spi89], this introduced a range of skills and concepts associated with formal development and verification. Additional core modules in the first two years covered aspects of discrete mathematics and logic which provided a good background. A further, optional module is offered to fourth year students on an MEng degree.

Changes to the curriculum were motivated by a number of worthy considerations, such as allowing students greater choice, but had the unfortunate (from our point of view) result of combining the second year logic and formal methods offerings into a single module. This posed the challenge of providing some meaningful and coherent coverage of formal methods in just a few weeks. Trying to pick out parts of the old module seemed unattractive. Any notation which involved too great a start-up time or which required familiarity with too much syntax would not work. Teaching a subset of syntax may result in insufficient knowledge of the language to be able to specify things correctly. Or worse, students are forced to invent convoluted ways to describe things that only reinforce a view of formal methods as being inaccessible.

This leads to the question of what can be achieved. What are the benefits of teaching formal methods and which, if any, can be derived from just half a term's study? One answer to this might simply be to provide an existence proof to show what formal methods are, how they are used, and why they are useful. However, we want students to be involved with "doing" and to experience for themselves the important skills and benefits of abstract modelling and reasoning, rather than racing to implementation. Many students arrive with an established procedural programming mindset. The growing use of rapid application development techniques are appealing to students who like the immediacy of such approaches. Encountering formal methods challenges preconceived ideas, particularly if the method allows students to spot errors which would otherwise have resulted in a flawed implementation. To

support this, it is useful to have a tool which can help reveal such errors and help users explore implications of the specifications they write.

Our approach has been to adopt Jackson's Alloy [Jac06] language, supported by the Alloy Analyser, sometimes referred to as a "lightweight" formal method. We have now used this approach for two successive cohorts of students.

3 Lightweight formal methods

The concept of a "lightweight" formal method is characterised by Jackson and Wing [JW96] as a method which has a formal basis but is limited in its scope by one or more of: partiality in language; partiality in modelling, partiality in analysis and partiality in composition. The term is used both for the "light touch" application of a traditional method and for methods which are designed specifically to cover only certain aspects of concern. A number of studies demonstrate the former approach, for example, in requirements analysis [ELC+98,AL98]. In formal methods education, Simpson [Sim06] refers to a "light touch" application to demonstrate the relevance of formal methods to professional software engineers.

The relationship between the different uses of the terms "lightweight" and "light touch" raises an interesting question. In this paper however we are concerned in particular with one notable approach specifically designed (and designated) as a lightweight method. Alloy [Jac06], inspired by the state-based Z notation [Spi89], allows the user to express complex structures and constraints using a relational language. The language is intended to be accessible and familiar (at least in part) to programmers used to object modelling with notations such as UML's Object Constraint Language [Gro03]. Properties of the model can be checked for a size of state space suggested by the user. A further lightweight method is Escher Technologies's Perfect Developer [Cro03].

One aim of using a lightweight approach is that a lighter touch can provide quicker (and hopefully automated) feedback on some properties of the system. Such feedback can inform the ongoing construction of a specification and the development of the system itself. In addition it may be hoped that a less ambitious method may well consist of a smaller language and limited user options so that the learning time required to start using the method could be less. Obviously, the partial nature of the method (or of the way the method is applied) means that the results have to be viewed in the light of the method's limitations. For example, a property which has been checked for a state space of limited size obviously does not have the same status as a property which has been verified for all possible values.

The limited nature of lightweight methods has led to some scepticism over their value. Boute characterises such a method as one which "attempts minimizing the user's exposure to mathematics, and therefore supposedly is more suitable to software engineers" [Bou03]. He warns that marketing these methods as accessible to users with little mathematical preparation may end in disappointing results which may discredit formal methods in general. However, Boute is more optimistic about the prospects for use in education: "For many, lightweight formal methods may be

Fig. 1. Alloy Analyzer

very valuable as a means to illustrate more abstract concepts and thereby lowering the threshold for the mathematics relevant to software engineering" [Bou03].

Our requirements were somewhat different from a desire to introduce "mathematics by stealth". Indeed, Alloy does not claim to do this. Our experience is that, to make progress, the user must understand the logical and relational framework of the notation and be able to use it to model at an abstract level. The attractions for us were the compact language, the relative speed with which small models can be written and explored and the guidance and feedback provided by small scope model checking. We consider below how far these are achieved in practice.

The Alloy notation supports modelling by *signature* and *constraint* definition, allowing quite complex structures. No particular methodology is enforced: Alloy can be used in a Z-like state+operations approach or in a variety of other ways to construct traces or describe (and solve) logical problems. Supporting the language is the Alloy Analyzer [3], a tool designed to explore models and check properties of interest. The tool uses SAT solving techniques to find, where possible, an instance of a given property which satisfies the definitions of the model. Limits must be placed on the size of the model to check. Existence of an instance demonstrates satisfiability, but failure to find an instance indicates only that one cannot be found in the current scope. The Analyzer therefore does not provide general results in the way that theorem proving can, but it does provide a quick, automatic way of exploring specifications and generating counterexamples. Jackson's "small scope hypothesis" [Jac06] postulates that most flaws can be detected in a small model.

There are obvious limitations to this approach but, with time constraints in mind, it seemed a possible way forward for students with some mathematical knowledge embarking on their first foray into formal methods. The language is relatively simple and tool support allows students to explore both the approach and the language itself but without being weighed down in large amounts of complex syntax.

We have now been teaching Alloy for two years. In addition to lectures, a good deal of teaching time is devoted to laboratory sessions in which the students can work through practical exercises, receiving help as necessary. Below, we reflect on the students' experiences. Information was obtained by talking to students; noting the questions they ask and the problems they meet; written feedback from module assessment forms and students' assessed work; observing students' interaction with the tool and the way they approach problem-solving.

4 Issues arising from students learning Alloy

This section identifies a number of issues commonly raised by students using Alloy.

One particular problem occurs in the shift from procedural programming. This is partly due to the young age at which many students start to explore computer

[3] Available from the Alloy website – `http://alloy.mit.edu`

```
pred addReg [s:Student, t:Tutor, c,c':Course] {
    s !in c.reg
    c'.reg = c.reg + s
    c'.alloc = c.alloc ++ s -> t
    c'.result = c.result
    }
```

Fig. 2. Alloy predicate

programming, reinforced by undergraduate programming modules. This leads to entrenchment in procedural programming. Students fail to appreciate the differences both in language terms and the shift in thinking required to model declaratively rather than functionally. For example, consider the predicate shown in figure 2 in which logical conjunction is implied between lines. Students view this predicate as a series of sequential operations where, somehow, `c'.reg = c.reg + s` occurs followed by `c'.alloc = c.alloc ++ s -> t`. Another example is the way in which the predicate relates the two courses `c` and `c'`. Rather than mathematically connecting the two courses, we have observed students treating `c'` as a modified version of `c`. That is, where they have omitted part of the connecting statement, students are surprised to find Alloy altering part of the course without telling them! These issues are not confined to Alloy, but the examples generated by the tool force users to confront misconceptions much earlier than with methods we have previously used. The difficulty students encounter with moving from procedural thinking indicate that it is an important consideration for any teaching approach.

Students are frequently observed to struggle with interpreting the visualisation output of the tool. Using the Alloy Analyser can be a somewhat frustrating process. There is certainly a degree of skill required in adjusting the display settings to avoid a confusion of spaghetti lines. Careful thought, examination and sometimes further experiment is required to understand the feedback provided. Efforts are underway to provide an automatic way of arranging the visualisation output in Alloy [RCD+07]. This includes improving the aesthetic qualities of the model and also, by examining the structural properties of the model, manipulating the display to ease understanding of complex results. Even when correctly specifying the model, students can encounter difficulties when examining the feedback from the tool. A common question from students is: "is it the correct answer?". Understanding the output from any tool can be an art in itself, particularly for any reasonably complex models. Further experiment can be required to interrogate the tool in a different way or to find further examples. With practice this improves but it takes time and encouragement to cultivate the spirit of enquiry and practical skills necessary to enable students to perform further meaningful experiments on their model.

The importance to students of tool support should not be underestimated. It implies a certain level of maturity in the language, a commitment to the techniques presented beyond that of an "academic exercise" and the ability to directly experiment with the approach not afforded with a paper and pencil exercise. The Analyzer leads students to develop a model incrementally and explore it as they go along. This is popular with the students and helps reveal many errors. Laboratory sessions are needed to support this. Issues relating to tool use are discussed below.

Another challenge commonly faced in formal methods education which has been

discussed elsewhere [RJ04] is that of motivation. Any curriculum which compartmentalises formal methods risks making this worse. This year, we have started to link the formal approach in small ways to material from other modules. For example, Alloy is well-suited to exploring logical puzzles and mathematical constructs (such as partial orders and lattices) which the students meet in other contexts. This has proved useful in making connections and introducing the tool as a useful aid (particularly with visualisation) before considering it in the context of specification. This was well-received by the students and is an area we plan to develop.

5 Issues of Students and Alloy

There is obviously a limit to what can be achieved in a short space of time no matter what method is used. Bearing this in mind, we were interested to see to what extent our aims for the module could be met.

Does the approach show formal methods as useful and relevant?
The response to this is positive from two separate aspects. Firstly, only a small amount of prerequisite material needs to be covered before some quite interesting case studies can be tackled. Even students who found the material challenging expressed a respect for the subject and were interested by what could be achieved in small case studies. Secondly, in their interaction with the tool, students could see that flaws in their thinking were being picked up and demonstrated to them at a very early stage of development. This again helps to show that formal methods can extend our understanding of a specification and catch errors that otherwise head unchecked into implementation. Whilst the possibilities are not as broad as with a full formal approach, we were encouraged by the positive response of the students.

Does the approach encourage abstract thinking?
Writing an abstract model challenges students' view of procedural behaviour. Perhaps it is more accurate to say that the feedback from the Analyser provides the challenge since, very often, simply writing the statements does not bring home the point that the model is much different to a program. It is interesting to observe how effectively a small amount of interaction with a tool can bring home points that we might talk about at length in lectures! Both the explanation and the exploration seem to be important here. Why abstraction is useful and what it allows us to do are much easier to address when students can connect to the ideas in practice. As an initial step to challenge students and present ideas of modelling and abstraction, Alloy has been as effective as a full formal approach, and perhaps even better due to the immediate feedback from the tool.

Does the approach hide the mathematics?
The model checking approach means that students do not have to battle with a proof system or understand how to direct a theorem prover. However, they work with an expressive notation and build fairly complex structures and relationships. However, this is not because the system is "hiding" anything from the user: it is simply finding instances. Obviously, we would want students to go on to study verification and refinement, but as a starting point, Alloy is quite honest in what it presents.

Another interpret ation of this question might be whether users can effectively write models without understanding the underlying logic and relational representation. Whilst the use of a programmer-friendly syntax may make statements appear more approachable to users, our experience is that to make any progress students need to understand the relational and logical levels of the notation and to become competent with abstract modelling. This perhaps means that the notation is not quite so user-friendly as we might like it to be. To use it, you have to understand it.

Does the tool help student learning?

A number of authors have suggested that it is not necessary for a formal method used in an educational setting to have tool support: indeed, it is also suggested that it may be better for students to work with paper and pencil. For example, Bayley *et al* [BLM06] raise the issue of students becoming discouraged by tools which reject their early efforts with cryptic messages. They also refer to the danger that users may take false assurance from a tool which checks only for syntactic correctness. These are real concerns, but balanced against them is the very apparent fact that, unchecked, it is all too easy to write rubbish. This is not confined to students and new users! Without feedback, the user's ideas are not scrutinised and challenged, and human appraisal may leave many errors undetected. Of course, proving properties of a formal specification is one way of exploring it and demonstrating that it has "correct" behaviour. However, many errors discovered when using Alloy relate to validation, that is, to building "the right program". By seeing examples, a user realises that they have not modelled the system they had in mind. Or perhaps, a requirement is only articulated as a result of behaviour viewed in the model. Verifying a property of a specification is only helpful if it specifies the system we want.

Another class of errors we observed relates to the work of Vinter *et al* [VLK98] which suggests that errors in mental constructions of certain logical expressions are extremely common, even amongst experienced users. This relates to Boute's comments [Bou03] on the way that false conclusions are commonly drawn. Vinter *et al*'s work confirms in a formal methods context some well-known psychological results which demonstrate that logical reasoning is subject to cognitive bias and systematic misconstrual. While modelling in Alloy is just as prone to this as any other formal method, the feedback from the analyzer does serve to confront the user with instances that reveal and challenge such misconceptions.

As discussed in the previous section, another useful aspect of the tool and the exploratory nature of using it is that students relate well to this way of working. This should not outweigh the question of what is appropriate educationally, but if the two are compatible then an approach which puts students at ease and which appears to them to be relevant and usable is an advantage. We have found that using the Alloy Analyser has been very beneficial in demonstrating the ideas from lectures and notes and the practical aspect is very important for effective learning (see Section 6). However, we do have some concerns about the way the tool is used.

Using a tool to provide feedback to inform the specification is very beneficial. However, we have also observed students replacing guided thought with blind trial and error. When a student reaches the point where they are not thinking about

the underlying specification but repeatedly trying out minor changes with no real understanding, the process changes from guided exploration to thoughtless hacking. Even if a "good" answer is obtained (such as no counterexamples found) the student really has little idea of what that means. This in turn can give rise to an unjustified faith in the model with the feedback from the tool misinterpreted. This is an area we would like to explore further to see if this is a significant problem and to understand at what point students abandon logic and how best to guide them.

6 Formal Methods Education

The way of working supported by Alloy allows small experiments to be performed on partial models, promoting exploration and interactive model-building. Properties of the model can be explored without having to specify a complete system. This is similar to the idea of "extreme specification" [Cro03] [4] in which functionality of a specification is developed incrementally and guided by feedback at each iteration. Examples (and counterexamples) that satisfy the current model can be generated. Then, as these examples are understood in the context of the given model, more complex and abstract ideas arise. Experiment and visualisation help shape the mathematical understanding of a model and give insight beyond an understanding of the mathematical symbols. This is similar to Lakatos' notion of experimental mathematics [Lak76] whereby understanding is developed through exploration and (failed) proof. With a traditional approach, students typically have a view of formal methods that is steeped in rule and rote, where the procedure is fixed and inflexible. We have observed that students can lose touch with the "reality" of requirements and connections to an end result, adopting a formalist approach in which they become focused on manipulating symbols. Experimentation and visualisation, for example, of the effects of a particular operation, can help to reconnect a student to the purpose of the task. In addition, formal insight often comes from playing informally with a model. Through small experiments that inform the developer's knowledge, the understanding of the model grows. These stages typically precede any further formal representation of the model. The search for the finished model is not blind but guided by experience and meanings extracted from interaction.

Students' use of Alloy can be seen as a particular kind of exploratory learning. Papert's constructionism [Pap93], founded on Piaget's constructivist ideas, argues that students learn best when actively engaging in building knowledge structures which can be discussed and further examined. Papert and Harel are careful to establish that constructionism is not simply "learning-by-making" [PH91]. Constructionism places special emphasis on the correspondence between "making in the world" supporting the mental processes of knowledge construction. Students are constructing a public "product that can be shown, discussed, examined, probed and admired" [Pap93, p. 142]. In practical terms, this means that a good deal of time is devoted to laboratory sessions in which students can carry out their explorations in an environment where help and direction is on hand. Model development

[4] Variously credited to Susan Stepney and Helen Treharne!

in Alloy is similar to the "bricolage" style of construction [LS68] where, as the model is built and evolves, so does the student's understanding of the situation. Testing of the model is integrated into the task of construction. The ideas embodied in a model are continually open for examination and, if necessary, revision.

Ideally, students develop a toolkit for tackling formal problems; providing them with the capacity to extrapolate beyond the information given and to ask questions of their model beyond the scope of the original question. The constructionist approach to learning is connected to motivation. Students must be willing and able to learn about formal methods. This only succeeds if students can see its worth and can engage with the material appropriately. If the students find some personal interest and stake in the material, they are more likely to gain a deeper and fuller understanding of the subject. This connects with some deeper issues of the role and place of formal methods in the Computer Science curriculum.

7 Conclusion and future work

This paper is certainly not an endorsement for taking curriculum time away from formal methods. Relegating them to a small corner of the timetable leaves little room to explore ideas fundamental to any Computer Science degree. Also, it makes these ideas appear unusual and less relevant to "normal" practice. The use of Alloy described here is as an introduction to students who have already had reasonable exposure to discrete maths and logic. There are many topics, such as verification and refinement, which we do not have time to explore and which would require moving beyond the bounds of the basic Alloy provision. Concerns also arise over the way that some students, particularly weaker ones, interact with the tool. Without engagement with the model and interpretation of feedback, a trial-and-error hacking approach can result. Occasionally, students appear to lose all sense of context and reality in an attempt to gain a formal tick of approval from the tool. It is also possible to gain a false sense of security by misinterpreting the tool's feedback. Despite these concerns, our overall experience has been very positive.

We may still strive for a situation in which formal methods pervade and inform the curriculum, but in the meantime a practical approach is needed. We have investigated one possibility - and therefore our findings apply to Alloy in particular rather than to other methods and approaches referred to as "lightweight". This has worked well for our circumstances. In fact, it is interesting to consider whether this approach would provide a good introduction even if time allowed a much fuller exploration of formal methods. Although partial in some aspects, it provides an approachable first encounter which is honest in its limitations and which can demonstrate a real usefulness in identifying flaws. Whether this is enough to attract more students to return for the later, optional formal methods module remains to be seen.

In this paper we have noted some of the benefits and limitations of the use of an interactive tool. This is an area we are keen to explore further. By making some minor alterations, we are planning to record information about how the tool is used and what steps a student takes in creating and improving a model. This will allow

both qualitative and quantitative analysis of aspects such as progress made towards a correct solution, iterations required, timing between checks.

References

[AL98] S. Agerholm and P. G. Larsen. A lightweight approach to formal methods. In *Proceedings of the International Workshop on Current Trends in Applied Formal Methods*, volume 41. Springer LNCS, 1998.

[Bac06] Roland Backhouse. Algorithmic problem solving - three years on. In *Teaching Formal Methods: Practice and Experience*, 2006.

[BLM06] Ian Bayley, David Lightfoot, and Clare Martin. Teaching the oxford brookes formal specification module. In J.P.Bowen P. Boca and D.A.Duce, editors, *Teaching Formal Methods 2006*, Oxford Brookes University, 2006. Available at: http://cms.brookes.ac.uk/tfm2006/.

[Bou03] Raymond Boute. Can lightweight formal methods carry the weight? In David Duce et al., editors, *Teaching Formal Methods 2003*, Oxford Brookes University, 2003. Available at: http://cms.brookes.ac.uk/tfm2003/.

[Cro03] David Crocker. Teaching Formal Methods with Perfect Developer. In David Duce et al., editors, *Teaching Formal Methods: Practice and Experience*, Oxford Brookes University, 2003.

[ELC+98] S. Easterbrook, R. Lutz, R. Covington, J. Kelly, Y. Ampo, and D. Hamilton. Experiences using lightweight formal methods for requirements modelling. *IEEE Transactions on Software Engineering*, 24, 1998.

[Gro03] Object Management Group. Object constraint language specification v. 2, 2003. Available from: http://www.omg.org/technology/documents/formal/ocl.htm.

[Jac06] Daniel Jackson. *Software Abstractions: Logic, Language and Analysis*. MIT Press, 2006.

[JW96] D. Jackson and J. M. Wing. Lightweight formal methods. *IEEE Computer*, April 1996.

[Lak76] Imre Lakatos. *Proofs and Refutations*. Cambridge University Press, 1976.

[LS68] C. Levi-Strauss. *The Savage Mind*. University of Chicago Press, 1968.

[Pap93] Seymour Papert. *The Children's Machine*. Basic Books, 1993.

[PH91] Seymour Papert and Idit Harel, editors. *Constructionism*, chapter 1. Ablex Publishing, 1991.

[RCD+07] Derek Rayside, Felix Chang, Greg Dennis, Robert Seater, and Daniel Jackson. Automatic visualization of relational logic models. In *First Workshop on the Layout of (Software) Engineering Diagrams (LED'07)*, 2007.

[RJ04] J.N. Reed and J.E.Sinclair. Motivating study of formal methods in the classroom. In *Teaching Formal Methods*, volume 3294. Springer, LNCS, 2004.

[Sim06] Andrew Simpson. Logic, damned logic and statistics. In J.P.Bowen P. Boca and D.A.Duce, editors, *Teaching Formal Methods 2006*, Oxford Brookes University, 2006. Available at: http://cms.brookes.ac.uk/tfm2006/.

[Spi89] J. M. Spivey. *The Z notation : a reference manual*. Prentice-Hall, 2nd edition, 1989.

[Tod05] USA Today. Fewer students major in computer. http://www.usatoday.com/tech/news/2005-05-22-computer-science-usat_x.htm, 2005.

[UCA07] UCAS. UK Universities and Colleges Admissions Service. http://www.ucas.ac.uk/, 2007.

[VLK98] R. Vinter, M. Loomes, and D. Kornbrot. Applying Software Metrics to Formal Specifications: A Cognitive Approach. *IEEE Metrics*, pages 216–223, 1998.

[Win00] Jeanette Wing. Weaving formal methods into the undergraduate curriculum. In *Proceedings of the 8th Int. Conf. on Algebraic Methodology and Software Technology*, volume 1816. Springer LNCS, 2000. Available at: www.cs.utexas.edu/users/csed/FM/docs/Wing-abstract.pdf.