Towards Increased Verification Automation for High Integrity Software Engineering

EPSRC Research Assistant Industrial Secondment

Final Report

Collaborative Training Account GR/T11289

Andrew Ireland

November 21, 2005

1 Introduction

We report here on the SPADEase project, a 6-month Research Assistant Industrial Secondment (RAIS) funded by the EPSRC Collaborative Training Account (GR/T11289). SPADEase followed on from NuSPADE, a 3-year project funded under the EPSRC Critical Systems Programme (GR/R24081) in collaboration with Praxis High Integrity Systems Ltd[1].

2 Background

Praxis High Integrity Systems Ltd (henceforth, Praxis) are an internationally leading developer of high integrity software systems, based in Bath, England. They advocate the SPARK Approach to software development, building upon the traditional Floyd-Hoare style of program reasoning [4, 5]. The SPARK Approach has recently been recognised by the National Cyber Security Partnership (NCSP) as one of only three techniques that can improve security critical software [8].

Central to the SPARK Approach is the SPARK programming language [1]. The SPARK programming language is defined as a subset of Ada which is expressive enough for industrial applications, but restrictive enough to support rigorous analysis throughout the development process. In particular, the SPARK language includes program annotations, allowing the programmer to specify the intended behaviour of their programs. The SPARK tool-set supports formal verification, *i.e.* proving that a program satisfies its specification.

[1]Previously, Praxis Critical Systems Ltd.

Key to making the SPARK Approach more attractive to industry is minimising the amount of user interaction required in performing formal verification. Praxis have had significant success in automating exception freedom proofs (absence of run-time errors) using the SPADE Simplifier, a special purpose automated reasoning tool. Unfortunately, two key verification problems remain. Firstly, although the SPADE Simplifier is very successful, it still fails to prove certain classes of problems. Secondly, it is often necessary to strengthen a program specification in order to complete proofs. The SPARK tools offer no support in automating the strengthening of program specifications.

The NuSPADE project aimed to build upon the success of the SPARK Approach by increasing the level of automated program reasoning. Our starting point was *proof planning* [2], an Artificial Intelligence technique for automated proof discovery. Proof planning uses high-level proof outlines, known as *proof plans*, to guide the search for proofs. A proof plan is structured in terms of proof *methods*, where each method is associated with a general purpose tactic. A key strength of proof planning is the proof *critics* mechanism [6, 7] that supports the automatic analysis and patching of failed proof attempts. As noted above, the discovery of program specifications, *i.e.* program properties, is also required for proof automation. Although program analysis supports the automated discovery of program properties, it does not identify which properties are necessary in order for a proof attempt to succeed. To overcome this, we used proof planning to constrain the program analysis. This was achieved via the proof critics mechanism, *i.e.* proof-failure analysis was used to generate constraints for use during program analysis. This integration of proof planning and program analysis provided a strong platform for developing automated program reasoning techniques. By exploiting this platform, the NuSPADE project successfully increased proof automation within the SPARK Approach. More generally, NuSPADE demonstrated the value of proof planning paradigm. Firstly, in terms of promoting the portability of automated reasoning strategies. Secondly, in terms of developing strongly integrated reasoning components, *e.g.* the productive use of proof-failure analysis within program analysis.

3 Project Overview

In general, the RAIS proposal aimed to transfer the results of the NuSPADE project into an industrial environment. This would involve communicating the core ideas within NuSPADE to an industrial audience and demonstrating that these ideas were industrially viable. In practise, we planned to meet these goals by developing an industrial version of NuSPADE, called SPADEase[2].

3.1 Report on Project Objectives

The proposal identified *core* tasks as well *opportunistic* tasks. Below we report on our results with respect to these tasks.

[2]The name signifies the importance of ease-of-use in industrial systems.

3.1.1 Core Tasks

The first core task was consolidation. NuSPADE was developed as a research system, focusing on hard automation problems. Consequently, NuSPADE lacks the robustness and accessibility expected of industrial tools. A significant amount of refactoring was required to transform the academic NuSPADE into the industrial SPADEase. Given the short project time, emphasis was placed on consolidating the proof planning aspect of NuSPADE. Key changes anticipated and subsequently implemented include:

- **Initialisation** - Launching NuSPADE on a problem requires manual initialisation. This setup includes deciding what rules to make available and what top level strategy to invoke. In SPADEase the initialisation process was fully automated, significantly simplifying the tool's interface.
- **Architecture** - The NuSPADE architecture was entirely reconstructed as four interacting systems. This greatly improved the clarity of the system and facilitated integration with the SPARK tool-set.
- **Integration** - The behaviour of NuSPADE, and some SPARK tools, was refined to achieve a seamless integration within the SPARK Approach.

Although the consolidation phase was originally scheduled for 4-months, in practise it took 5-months. A small portion of the delay can be attributed to taking advantage of unforeseen training opportunities. However, primarily, the additional time was required to resolve unexpected technical obstacles. In particular:

- **Performance** - During development, some obvious extensions became apparent that would bolster the performance of SPADEase. The general proof strategy was slightly altered to accommodate a new collection of efficiency boosting methods. Further, NuSPADE's simplistic waterfall execution model was replaced with an explicit strategy mechanism. This supports increased control of the search space, allowing expensive but uninteresting search paths to be automatically curtailed.
- **Middle-Out Reasoning** - A powerful feature of proof planning is *middle-out reasoning* (MOR) [3], *i.e.* the use of meta-variables in delaying choice during the search for proofs. The NuSPADE implementation of MOR was not robust, so required re-implementation within SPADEase.
- **Simplification** - An unexpected side effect of automated initialisation in SPADEase was the effective failure of the NuSPADE simplification techniques[3]. This problem was addressed via the development of a new, more constrained, goal based simplification method.

The second core task was the evaluation of SPADEase. This was originally scheduled for 2-months, but had to be completed in 1-month as a result of delays in building SPADEase. Nevertheless, SPADEase was tested on a range

[3]They became *extremely* time consuming.

of examples. In addition, a more extensive evaluation against a large industrial example was also undertaken.

3.1.2 Opportunistic Tasks

Given the time constraints, it was unrealistic to consider the full industrialisation of NuSPADE. Thus, the scope of the project was restricted to the core tasks considered above. However, it was speculated that there might be opportunities to make progress on two additional tasks outside the core remit.

The first opportunistic task was the exploration of more advanced proof methods. In practise, however, industrial problems tend to require robust, controlled, reasoning rather than sophisticated reasoning. The introduction of new efficiency boosting methods and the necessary development of a new simplification method reflects this shift in emphasis.

Nevertheless, following the evaluation of SPADEase, proof patterns were discovered which suggested new proof methods for dealing with case splitting and goals involving enumeration types. While time prevented an implementation and deeper analysis of these proof methods, it was encouraging to note that SPADEase assisted in the identification of new candidate methods.

The second opportunistic task was to investigate incorporating program analysis features into SPADEase. The SPADEase architecture was developed to actively support the capacity for program analysis. In addition, progress was made on explicitly identifying the key changes required to extract industrial program analysis from NuSPADE. However, on identifying these tasks, it became clear that there was not sufficient time to complete on this.

3.2 Key Outcomes

SPADEase can be invoked on a SPARK problem with a single command. This reflects the behaviour of a genuine industrial tool, and is a significant improvement over NuSPADE. Proof search, especially on non-trivial problems, will always be time consuming. While SPADEase is no exception to this rule, it makes positive steps toward addressing this concern through various efficiency saving techniques. By building on the proof planning paradigm, SPADEase presents a high level view of proof automation. This style of automation is particularly attractive to industry as it is logically sound while remaining both transparent and immediately extensible. Furthermore, as proof planning captures the key intuitions behind proofs, its failure can be especially revealing, highlighting areas where reasoning might be strengthened.

As SPADEase lacks a program analysis component it is unable to automatically strengthen specifications. This shortcoming was evident during the evaluation, leading to a scattering of problems that NuSPADE could automatically discharge yet SPADEase could not. Nevertheless, this weakness serves to underline the additional automation leverage that can be gained from program analysis.

4 Training and Related Activities

The RA, Mr Bill Ellis, was also the RA on the NuSPADE project. As well as developing SPADEase, Bill also took part in a number of training and related activities:

- Attended a "SPARK Black Belt" course – an advanced course for SPARK software engineers.
- Praxis has a strong research oriented culture, so Bill presented two seminars during his secondment as well as attending other technical/research talks.
- The NuSPADE project involved significant work with SICStus Prolog. Bill's presence at Praxis coincided with their adoption of SICStus as the main Prolog for the SPADE proof tools. Thus Bill was able to assist in the resulting porting process.
- Bill also learned about many of the low-level support tasks associated with commercial software development, *e.g.* version control, regression testing, bug tracking, *etc.*

Bill's industrial secondment inevitably slowed down the writing-up of his PhD. However, the work which he undertook on the secondment will enrich his final thesis. Bill he is now writing-up full-time and is expecting to submit in the early part of 2006.

5 Research Output

A major output from the project was the SPADEase tool. A conference paper describing both the NuSPADE and SPADEase tools is in preparation. This paper will have a technology transfer component. In addition, the results of the SPADEase project are being integrated into a journal length paper for submission to the International Journal of Automated Reasoning, special issue on "Empirically Successful Automated Reasoning" – submission deadline, Dec 5^{th} 2005.

6 Dissemination Activities

As well as the papers highlighted above, web pages for the SPADEase project have been produced, *i.e.*

`http://www.macs.hw.ac.uk/spadease/`

In terms of advertising the success of SPADEase, Praxis provided the following endorsement:

"SPADEase represents a very significant advance in the practical application of proof planning. It increases the proportion of SPARK-generated verification conditions that can be proved automatically without introducing any new opaque, black-box processes. The separation of proof planning from proof checking also acts as a talent multiplier by allowing proof experts to spend their time creating new and reusable methods and approaches rather than working on individual proofs."

Peter Amey, Chief Technical Officer,
Praxis High Integrity Systems Ltd.

7 Project Management

Given the nature of the secondment, project management was relatively light touch. Supervision was essentially by email, and the School of Mathematical and Computer Sciences (Heriot-Watt) funded one mid-term review meeting which was held at Praxis in Bath. Within Praxis, Bill's day-to-day supervision was undertaken by Dr Rod Chapman.

8 Follow-on Support

We see SPADEase as a first step towards technology transfer. We are actively looking to deploy and further enhance the tool within a *live* development project. Linked with this is a collaborative proposal that is currently being evaluated by ITI Techmedia. The proposal focuses on *Automated Security Engineering* and is in collaboration with the School of Informatics (University of Edinburgh). One of the aims of this collaboration is to extend SPADEase so that it provides a "bridgehead" for our on going automated reasoning within the formal methods market. In addition, the Defence Science and Technology Laboratory (Dstl) have shown interest in supporting SPADEase in terms of technology transfer. To this end, Dstl and Praxis are currently looking to identify a MOD programme which would benefit from the use of SPADEase. If successful, such a project would start in Spring 2006.

References

[1] J. Barnes. *High Integrity Software: The SPARK Approach to Safety and Security.* Addison-Wesley, 2003.

[2] A. Bundy. The use of explicit plans to guide inductive proofs. In R. Lusk and R. Overbeek, editors, *9th International Conference on Automated Deduction*, pages 111–120. Springer-Verlag, 1988. Longer version available from Edinburgh as DAI Research Paper No. 349.

[3] A. Bundy, A. Smaill, and J. Hesketh. Turning eureka steps into calculations in automatic program synthesis. In S. L.H. Clarke, editor, *Proceedings of UK IT 90*, pages 221–6. IEE, 1990. Also available from Edinburgh as DAI Research Paper 448.

[4] R. W. Floyd. Assigning meanings to programs. In J. T. Schwartz, editor, *Mathematical Aspects of Computer Science, Proceedings of Symposia in Applied Mathematics 19*, pages 19–32. American Mathematical Society, 1967.

[5] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12:576–583, 1969.

[6] A. Ireland. The Use of Planning Critics in Mechanizing Inductive Proofs. In A. Voronkov, editor, *International Conference on Logic Programming and Automated Reasoning (LPAR'92), St. Petersburg*, Lecture Notes in Artificial Intelligence No. 624, pages 178–189. Springer-Verlag, 1992. Also available from Edinburgh as DAI Research Paper 592.

[7] A. Ireland and A. Bundy. Productive use of failure in inductive proof. *Journal of Automated Reasoning*, 16(1–2):79–111, 1996. Also available as DAI Research Paper No 716, Dept. of Artificial Intelligence, Edinburgh.

[8] National Cyber Security Partnership (NCSP). Improving security across the software development lifecycle. 2004. http://www.cyberpartnership.org.