INTRODUCTION TO FORMAL METHODS AND EXPERIENCES FROM THE LACOS AND ØRSTED PROJECTS

J. Storbank Pedersen[1]

Introduction

The primary feature that distinguishes formal methods for systems development from other development methods is the use of a formal notation for describing the system to be developed. The main reasons for using a formal notation are that the descriptions produced are unambiguous and that they can be subject to reliable reasoning about the system's behaviour prior to implementation. In addition to the precision, a notation used by a formal method includes facilities for writing specifications that are more abstract than programs, in order to focus on *what* a system should do rather than on *how* it is done. Moreover, most formal methods recommend a stepwise development approach where increasingly more concrete formal specifications are produced and where development relations between concrete and corresponding abstract specifications can be formally expressed and potentially proven.

In fact different levels of formality can be applied when using a formal method. These levels include:

- Formal specifications only, but no formal relations or proofs
- Formal specifications and formal expression of development relations, but no proofs
- Formal specifications and formal expression of development relations, and informal justifications of the properties
- Formal specifications and formal expression of development relations, and fully formal proofs.

Each project that uses a formal method should choose its level of formality based on for example resource constraints, as it might not be feasible to completely formally prove the correctness of a large system.

One of the main reasons for using formal methods is that they improve the quality of the systems being produced. But improved quality cannot necessarily always justify the costs of the improvement. However, formal methods have been used for several years and on a wide variety of projects, and mature commercial tools supporting the use of formal methods are available. In particular the collective experiences gained and documented by projects that have used formal methods could help future users in making informed decisions regarding where and how it is feasible to use formal methods.

The LaCoS project

The LaCoS project was an ESPRIT project that was carried out between 1990 and 1994. Its aims were to conduct extensive industrial trials using the formal method RAISE (Rigorous Approach to Industrial Software Engineering) within a number of industrial sectors, and to further evolve and industrialise the RAISE products, i.e. the RAISE Specification Language (RSL), the RAISE method and the RAISE tools.

The partners of the project were divided into *producers* and *consumers*; the former were CRI (DK), BNR (UK) and CAP Programator (DK).

---

[1] J. Storbank Pedersen is with CRI A/S, Birkerød, Denmark

The applications covered a wide range of areas, and the main ones were:

- A concurrent database (Bull, F)
- Cryptographic protocols (Bull, F)
- A network design tool (BNR, UK)
- An automatic train protection system (MATRA Transport, F)
- Security protocol verification (MATRA Transport, F)
- Ship engine monitoring (Lloyd's Register, UK)
- A car engine management system (Lloyd's Register, UK)
- A security model (Lloyd's Register, UK)
- Software for a tethered satellite (Space Software Italia, I)
- An Air traffic control System (Space Software Italia, I)
- An image processing system (Inisel Espacio, ES)
- A shipping transaction processing system (Technisystems, GR)

The applications where the use of RAISE was found to be most successful were the ones having at least one of the following properties, (1) there were particularly high quality requirements and hence significant properties were formally justified, (2) the system had an inherent complexity that would benefit from formal specification, (3) the system could be produced by automatic translation from formal specifications, or (4) an existing system had to be analysed and reverse engineering (abstraction) from code to formal specifications was applied as the means of analysis.

One important lesson learned was that even though new users of a formal method go through some initial training before starting to use such a method, they need to have access to an expert in the method in order to get timely and qualified feed-back on the specifications being produced. This expertise can either be made available in-house or by external consultants.

It turned out that RAISE, like most other formal methods, fitted well into the existing software life-cycles used by the various companies. This was seen as a major benefit because introducing formal methods would be considered by both managers and developers as an evolution rather than a revolution.

It was found that using a formal method generally led to more time spent in the specification and early design phases compared to other methods. But it also led to an earlier discovery of errors and identification of difficulties in the systems.

The Øersted project

The Øersted satellite is a Danish micro-satellite (c. 60 Kg.) whose mission is to measure the magnetic fields and fluxes of charged particles near the Earth. The satellite carries five scientific instruments including an Overhauser magnetometer, a fluxgate magnetometer and a star imager, all these three instruments being placed on a deployable 8 metre boom. The development is now complete and the satellite is to be launched in the autumn of 1997.

The on-board software that was developed by CRI constitutes two command and data handling sub-systems and consists of approximately 60,000 lines of Ada code. The command and data handling sub-systems are responsible for up and down link telecommunication, acquisition and pre-processing of scientific data, failure detection and recovery based on housekeeping collection, and autonomous control of data acquisition. The software is hence critical to the mission.

It was decided to divide the considerations of functional and performance correctness, and to use RAISE as the primary method for achieving the former. In order to be able to test the satellite, including software, at various

points during the development, the software was incrementally developed, with each new increment representing increased functionality. In order to handle the real-time aspects a combined method approached was used where:

- HRT-HOOD (Hard Real-Time Hierarchical Object Oriented Design) from ESA was used for the architectural design and for expressing the real-time properties.
- RAISE was used for expressing the functionality in the detailed design
- Ada was used for coding

One of the reasons for using RAISE was to be able to use the RAISE tools' ability to automatically produce Ada code from formal specifications. This was particularly important because of the incremental development approach that meant that five increments had to be delivered for test within a tight time schedule. On-board software typically has stringent requirements on both size and performance of the code. The Ada code was produced automatically for the sequential parts of the RAISE design, and by hand for the concurrent parts and certain critical composite data structures.

The main conclusions from the software development were that it was indeed feasible to combine automatically generated code and hand-written code to produce on-board software, and that it was possible to automatically generate approximately 60% of the code for a command and data handling system from formal specifications.