Assessing the Formal Development of a Secure Partitioning Kernel with the B Method

André Passos, José Miguel Faria, Simão Melo de Sousa
Critical Software, DI - Universidade da Beira Interior

This extended abstract reports on the introduction and the use of a formal development cycle in the partial development of a real industrial application, namely the design of security policies enforcement mechanisms in real time operating systems, in particular using the "Correct By Construction"paradigm. The work allowed the exploitation of formal methods applied to such a complex system as a kernel based application. The main goal of this publication is to share the key results and conclusions achieved, specially focusing the issues related to the industrial adoption of formal methods.

The design and implementation of safe and secure systems has been proven to be a hard task to accomplish, especially when they rely upon the features of an underlying operating system. The Multiple Independent Levels of Security (MILS) was created to enable applications to enforce their own security policies. Rushby, when presenting the idea for the first time, in the early 80's, proposed that a separation kernel should divide memory into partitions and allow only carefully controlled communication between non-kernel partitions. MILS architecture provides the basis for the construction of a secure partitioning kernel. MILS also allows a remarkable reduction of the amount of security-critical code making it easier to inspect the respective code. Therefore, it is a perfect target for rigorous inspection and mathematical proof of correctness by techniques such as formal methods.

In this work, the B Method is used for the (partial) formal development of a secure partitioning kernel (SPK). This constitutes a novelty in the formal methods community, posing an extra challenge in the project: to evaluate the suitability of applying the B Method outside its usual application domain (railway and automotive). The B Method covers all the steps of the software development life cycle and is supported by a set of mature tools that allow the proof of correctness of the developed models and provide the desired functionality of automatic code generation.

The work performed can be divided in two phases. Initially, a complete development of a high-level model of the SPK was built. Then, part of the kernel was refined to a level where C code could be automatically generated from it. (This is called implementation level in B.) The high-level models of the kernel constitute a complete architectural design of the system. The ProB animator and model checker was used to animate and validate the models. ProB allows the analysis of the system behaviour by interactively generating all the possible instances of execution. Additionally, it was possible to observe that the use of an animator can be very successful when used as a communication tool between the different parts involved in a project, such as domain experts and B practitioners.

The validated high-level models could be refined for a completely formally developed SPK. In our work, and as a first step to this task, the Partition Information Flow Policy (PIFP), which is part of the SPK, was implemented. Information flow between partitions has to be monitored and controlled by the kernel. The PIFP defines the rules for each flow. Every time a partition tries to communicate with another, the SPK first checks (using the PIFP) whether or not the flow involving the two partitions is possible. With this mechanism, the SPK assures information security. The refinement process that led to the implementation of the PIFP was carried out with the assistance of Atelier B. This tool provides the necessary means to achieve early validation though the use of formal proof. Atelier B was also used for the generation of the C code. Indubitably, such capability is judged very important. It allows the reduction of the time spent writing and validating the implementation code (no unit and integration tests are made) while, at the same time, providing a continuity in the development process and assuring that the generated code correctly implements the specified functionalities.

At the final stage, an open source micro kernel – PREX – was adopted to integrate the proposed PIFP. With this, two goals were achieved: firstly, PREX provides the necessary evidences for testing the PIFP. Second, this procedure demonstrated the feasibility of applying formal methods only to parts of the system, since hand-written code could be coalesced with the automatically generated code.

Verification and validation has been achieved trough the use of tests and mathematical proof of correctness. In terms of formal proofs, the Atelier B automatically proved 88% of the generated proof obligations (which is a usual percentage in traditional developments using B). The remaining 12% are in the process of being interactively proved. Compared to the other activities in the project, this is the part which requires the highest proficiency in the methodology.

These first results show that the design and implementation of safe and secure system using Formal Methods in industry is possible and rewarding. Our current goals are related to the improvement of the proof effort and the further refinement of the remaining components.