DOMAIN-SPECIFIC LANGUAGES FOR REQUIREMENTS MODELLING (WITH A FOCUS ON IMA SEPARATION KERNELS) EXTENDED ABSTRACT

Andrew Butterfield[1] **and Ciaran Costello**[2]

[1]*Lero, Trinity College Dublin,Andrew.Butterfield@scss.tcd.ie*
[2]*Trinity College Dublin*

Presenter: *Andrew Butterfield*

1. BACKGROUND

The project *IMA-SP Kernel Qualification Preparation (IMAKQP)* an activity led by SciSys Ltd, involving CNES, Airbus DS, TASF, looking at the approach to be adopted for IMA-SP Separation Kernel qualification. The activity *Formal Methods Expert to IMA-SP Kernel Qualification Preparation (FMEIMAKQP)* [1] was a parallel joint activity involving Lero@TCD, exploring the potential role that formal verification techniques might play in a future kernel qualification process. It followed on from the experience of a previous activity *Method and Tools for Onboard Software Engineering (MTOBSE)* [2]

The MTOBSE project, involving two partners in Lero: the Irish Software Research Centre, namely the University of Limerick and Trinity College Dublin, looked at a top-down approach to formalising and verifying an IMA TSP kernel, and was reported on at two previous DASIA events[BSH13, BSH13]. We developed a reference specification for a Time-Space Partitioning (TSP) kernel[BS13a] and did a survey of formal techniques that had potential for verifying a kernel implementation[BS13b, BS13c]. We then selected Isabelle/HOL[NPW02] and used this to formalise the reference specification and some of the XtratuM code[BS13d]. We then reported on our experiences[BS13e]. A key focus of the MTOBSE project was to consider formally verifying a kernel to the extent required by the Common Criterion (CC) guidelines for both functional correctness and security[Cri05], with particular reference to the Separation Kernel Protection Profile (SKPP)[Dir07].

The focus of FMEIMAKQP was exploring approaches for qualifying an *existing* separation kernel implementation as its starting point. In particular, the focus was on functional correctness rather than security properties. From a formal perspective this involved looking much more closely at techniques for reasoning at or close to source code level. Also of interest was to explore how formal techniques might best be integrated with well-established software engineering practises, such as testing, and model-based development. Results of this activity were reported at DASIA in 2015 [BH15], and in 2016 [BCHH16].

2. KEY FMEIMAKQP OUTCOMES

Key formal-methods results in the Final Report for FMEIMAKQP[But16] where:

1. A High-Level model written in CSP[Hoa85] of the Leon-processor based platform, and some of the Requirement Baseline[Cor15] behavioural specifications. Analysis of these models was done using the model/refinement checking tool FDR3[GRABR14]. Some of this reported on work done in Trinity College Dublin for a Masters Thesis[Hen16]
2. The use of the Frama-C toolkit[C$^+$15] to add low-level annotations to the C sources for the Xtratum hypervisor[CRM10].

A key issue that has to be addressed for both of these results was the fact that they involved a consideration of "unavoidable concurreny", along with the non-determinism that usually arises. The purpose of the separation kernel is to allow multiple (software) partitions to run on one hardware platform, each being isolated in both time and space, with any cross-communication limited to what is authorised. For a single-core system, despite the fact that at any point in time, we have only one partition or the kernel itself actually running, we find that we have what is an essentially *concurrent* system. The CPU executes instruction in a sequential manner on behalf of either the kernel or partition. Also running concurrently is the MMU/MPU which observes the bus traffic, and checks, against its current configuration setup, if the

[1] funded by ESTEC CONTRACT No. 4000112208 supported, in part, by Science Foundation Ireland grant 13/RC/2094

[2] funded by ESTEC CONTRACT No. 4000106016, and supported, in part, by Science Foundation Ireland grant 10/CE/I1855

memory requests are permitted. Meanwhile, the interrupt request hardware, although part of the CPU, is effectively also running in parallel, taking in interrupt requests from IO devices and timers, as well as the MMU/MPU.

Key formal methods-related recommendations in [But16] include:

- Formalising the Requirements Baseline
  It would describe a relationship between a valid kernel configuration and the permissible behaviours associated with both the kernel and all the partitions defined in the configuration. The primary value of this model is that it would provide a way to check the requirements baseline for internal consistency.
- Formalising the Data Invariant
  A configuration invariant would allow the formal verification of configuration tools. It could also be used to verify the correct use by the kernel of its own internal data-structures.
- Verifying **PK-9**
  This is a requirement regarding the correct save/restore of partition state as a result of context-switches. It proved difficult to test adequately.
- Completing the CSP models
  This work [Hen16] was very much a proof of concept. But is has potential to do much more, particularly because CSP's model of events and concurrency is a good fit to the concurrent behaviour of the kernel platforms, and the general behavioural flavour of many of the requirements..

We also suggested other kinds of space-related modelling and software activities that might benefit from early formal modelling: e.g. SAVOIR (architecture,specifications,data models,interfaces, protocols).

The last recommendation discussed various modalities for future activities in this space. In particular it was noted that student projects are a way to conduct explorations of various ideas, as demonstrated by the work reported at DASIA 2016.

3. CURRENT INVESTIGATIONS

In this paper we will describe results of current investigations by Lero at Trinity College Dublin that explore how formal techniques could be used to support the verification of seperation kernel software and its related design artefacts, from requirements down.

3.1. Revising the CSP Model

Work is ongoing to transform the CSP models described in [Hen16] mainly to try to avoid the problem of "combinatorial state-space explosion" that bedevils model-checking based techniques. Some initial progress towards this goal was made in the thesis, but there is plenty of scope for improvement.

There are two lines of attack we are bringing to bear on this problem:

1. Abstraction: In order to be able to successfully run checks, we need to tale particular care to abstract away from all irrelevent aspects of system state. For example, in order to model how the MMU monitors memory accesses and how the kernel configures the MMU and handles its interrupts, we probably don't need to distinguish between reads/writes, or the specific data values, at least for many requirements. The challenge is find the greatest abstract that does not information that is in fact important for the requirement or behaviour being modelled.
2. Modularity: The verification approach is to take a model of the Kernel, some Partitions, and the Hardware Platform (KPHP), and construct a way to use the requirement model to act as a monitor for the kernel behaviour. With CSP it is possible to build individual models for each of (most of) the baseline requirements This makes it possible to check each requirement individually, which clearly a very useful aspect of this modelling approach. However much of this advantage is lost if a check for a given requirement fails because of state-space explosions due to parts of the KPHP state that are irrelevant to that given requirement. We are exploring how to (re-)factor the various models into sub-models that focus on key subsets of the overall space, so that each requirement check is tailored to only use the relevant subset.

3.2. A Requirements DSL

A student project is also investigating the development of a machine-readable domain-specific language (DSL) for requirements that can express the kinds of requirements we find in the kernel baseline. Such a language could then be used to develop tools to perform requirements consistency checking, and to connect the requirement to other modelling notations and techniques.

Some aspects of this are quite novel for the aerospace industry: we propose the implement the DSL using the functional programming language Haskell (www.haskell.org). This is because DSL development is something for which that language is ideally suited. The concrete language syntax is that of Haskell, so no parsers need be written. The language elements can be defined as atomic building blocks used in conjunction with combining forms, using the type-class system of Haskell. This leverages the type system to ensure descriptions are well-formed, but does not impose any *a-priori* semantics on the language. The class system is enacted by defining

various datatypes that do have semantic significance, and showing how they are instances of the DSL language. This means we can easily give the same description multiple meanings. Such meanings could comprise:

- A Text Rendering — useful for review of the requirements
- Structure Graphs — tailored for some kind of consistency/completness checking
- CSP Output — so we can exploit the CSP models and the FD3 model-checker.
- Simulator Output — generate something for an appropriate simulation tool.

The key challenge here is the correct identification of the atomic building blocks and combining forms.

REFERENCES

[BCHH16] A. Butterfield, Alexandre Cortier, Kevin Hennessy, and M. Hinchey. Towards formal verification of interrupts and hypercalls. In *DASIA 2016, The International Space System Engineering Conference*, Tallinn, June 2016. ASD-Eurospace.

[BH15] A. Butterfield and M. Hinchey. Towards the adoption of formal techniques for kernel qualification. In *DASIA 2015, The International Space System Engineering Conference*, Barcelona, June 2015. ASD-Eurospace.

[BS13a] A. Butterfield and D. Sanan. D03: Reference specification for a partitioning kernel and a separation kernel. Technical Report Version 8.1, Lero, 2013. ESTEC Contract No. 4000106016, comprising 3 documents: SRS, ADD, ICD.

[BS13b] A. Butterfield and D. Sanan. D04: Task 3: A formal verification process: approaches; feasibility; cost. Technical Report Version 4.0, Lero, 2013. ESTEC Contract No. 4000106016.

[BS13c] A. Butterfield and D. Sanan. D06: Evaluation of formal language, formal methods and modelling languages suitable for implementing an abstract specification. Technical Report Version 1.0, Lero, 2013. ESTEC Contract No. 4000106016.

[BS13d] A. Butterfield and D. Sanan. D07: Abstract Specification. Technical Report Version 4.0, Lero, 2013. ESTEC Contract No. 4000106016.

[BS13e] A. Butterfield and D. Sanan. D08: Assessment of the formal verification process. Technical Report Version 2.0, Lero, 2013. ESTEC Contract No. 4000106016.

[BSH13] A. Butterfield, D. Sanan, and M. Hinchey. Towards formal verification of a separation microkernel. In *DASIA 2013, The International Space System Engineering Conference*, Porto, May 2013. ASD-Eurospace.

[But16] A. Butterfield. R1: Formal methods expertise—final report. Technical Report FMEIMAKQP FR, Version 1.0, Lero@TCD, 2016. ESTEC Contract No. 4000112208.

[C+15] Loïc Correnson et al. *Frama-C User Manual*. CEA List, Software Safety Laboratory, Saclay, F-91191, release magensium-20151002 edition, 2015. `http://frama-c.com/download/frama-c-user-manual.pdf`.

[Cor15] Alexandre Cortier. D02: Partitioning Kernel Requirements Baseline. Technical Report Version 1.2.0, AIRBUS Defence and Space SAS, 2015. ESTEC Contract No. 4000111495/14/NL/GLC/al.

[Cri05] Common Criteria. Common criteria for information technology security evaluation, version 2.3. Technical Report ISO/IEC 15408:2005, Common Criteria, 2005.

[CRM10] Alfons Crespo, Ismael Ripoll, and Miguel Masmano. Partitioned embedded architecture based on hypervisor: The xtratum approach. In *Eighth European Dependable Computing Conference, EDCC-8 2010, Valencia, Spain, 28-30 April 2010*, pages 67–72. IEEE Computer Society, 2010.

[Dir07] Information Assurance Directorate. Protection profile for separation kernels in environments requiring high robustness. Technical report, U.S. Government, June 2007.

[GRABR14] Thomas Gibson-Robinson, Philip Armstrong, Alexandre Boulgakov, and A.W. Roscoe. FDR3 — A Modern Refinement Checker for CSP. In Erika brahm and Klaus Havelund, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 8413 of *Lecture Notes in Computer Science*, pages 187–201, 2014.

[Hen16] Kevin Hennessy. Investigating Interrupts in the Xtratum Hypervisor using CSP. Master of science in computer science (mcs), Computer Science and Statistics, Trinity College Dublin, O'Reilly Institute, Trinity College, Dublin 2, Ireland, May 2016.

[Hoa85] C. A. R. Hoare. *Communicating sequential processes*. computer science. Prentice-Hall International, Englewood Cliffs, N.J, 1985.

[NPW02] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.