An approach to
Integrating Software Models via Refinement

Marie Farrell mfarrell@cs.nuim.ie

Principles of Programming Research Group, Department of Computer Science,
National University of Ireland Maynooth

1. Motivation

In contrast to other engineering disciplines software design is by nature evolutionary. Features are often added and removed at the discretion of the project manager without thorough examination. This negligence can lead to both financial and human loss. For example, in 1996 the Ariane 5 launcher was discovered to have a software error which caused it to explode only seconds after launch rendering losses of approximately €350 million [Kumar et al. 2013; Charette 2005]. This failure could have been prevented if procedures to systematically engineer and verify this system had been followed. In fact, software bugs cost the global economy approximately $312 billion annually [Britton et al. 2013].

A formal specification is the exact definition in mathematical notation of what a system is required to do (and not do). In software development, it is common to model software at different levels of abstraction, starting with a very high level abstract specification and finishing with a detailed concrete implementation. In formal software engineering, we can map between these levels of abstraction in a verifiable way through a process known as refinement. In contrast to the approach used in model-driven engineering, refinement typically happens within a single modelling language.

2. Project Context:

This project is centred on the Event B formal specification language [Abrial, 2010] which is used in the verification of safety critical systems such as air traffic control and automated trains. The trademark example of Event B in action is its success of Paris métro line 14 where Event B was used to model the behaviour of the automated trains on line 14. Event B models are an instance of the specification and are made up of contexts and machines; contexts specify static properties of the model and machines describe the dynamic aspects of the model. Event B uses refinement to map between models with different levels of abstraction.

3. Aims

The goal of this project is to fully integrate this process of formal refinement into the model-driven engineering approach so that models expressed in different modelling languages, formalisms and tools can be combined within one formal framework in a provable correct way. The resulting framework will support the sharing of refinement steps, and their associated proofs, between different modelling environments.

The central research aim of this proposed work is to establish a theoretical framework within which refinement steps, and their associated proof obligations, can be shared between different formalisms. Our core hypothesis is that the theory of institutions [Goguen and Burstall, 1992] can provide this framework. Thus, we propose to construct an institution-based specification of the Event-B formalism.

4. Impact

The impact will be an improved software development process that allows the integration of software models, which focus on different aspects of the software, modelled at different levels of abstraction. The overall consequence will be the provision of a solid mathematical foundation for model-driven engineering.

5. Approach

First, specifications in Event-B will be able to interact with formalisms already in the institutional framework. Second, Event-B will bring new perspectives to institutions, based on its well-developed refinement model. Finally, by integrating an industry-strength approach into this framework we will be much closer to establishing a solid, formal foundation to the theory of model-driven engineering.

6. Work to Date

A collection of case studies have been constructed and verified using Event B and the refinement process.

References


