An approach to

Integrating Software Models via Refinement

Marie Farrell

Supervisors: Dr. Rosemary Monahan & Dr. James Power

Principles of Programming Research Group, Dept. of Computer Science,

National University of Ireland Maynooth

Software Errors

- Software design is by nature evolutionary. This means that features are added and removed at the discretion of the project manager, often without thorough examination.
- Ariane 5 launcher exploded due to a software error and cost an estimated **€350,000,000** [Charette, 2005].

- A software error caused the Therac-25 machines to give massive overdoses of radiation to six cancer patients. Some received over 100 times the required dosage. This excessive radiation exposure resulted in severe injuries and three patients' deaths [Kumar et al, 2013].

- Software bugs cost the economy $312 billion annually [Britton et al, 2013].

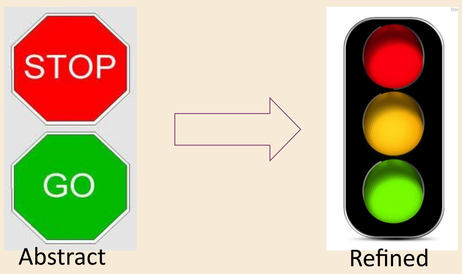
Formal Refinement

- A formal specification is the exact definition in mathematical notation of what the system is required to do (and not do).
- The Event B formal specification language is used in the verification of safety critical systems [Abrial, 2010].

- Event B models are an instance of the specification.

Machine	Context
variables	carrier sets
invariants	constants
events	axioms

- Refinement provides a way for us to model software at different levels of abstraction [Abrial et al, 2006].



Abstract Refined

Social Network Specification in Event B

```
MACHINE
    mac1
SEES
    ctx1
VARIABLES
    person
    rawcontent
    content
    owner
INVARIANTS
    inv1:   person ⊆ PERSON not theorem
    inv2:   rawcontent ⊆ RAWCONTENT not theorem
    inv3:   content ∈ rawcontent ↔ person not theorem
    inv4:   owner ∈ rawcontent → person not theorem
EVENTS
    INITIALISATION:    not extended ordinary
    THEN
        act1:   person ≔ ∅
        act2:   rawcontent ≔ ∅
        act3:   content ≔ ∅
        act4:   owner ≔ ∅
    END

    transmit:   not extended ordinary
    ANY
        rc
        pe
    WHERE
        grd1:   rc ∈ rawcontent not theorem
        grd2:   pe ∈ person not theorem
        grd3:   rc ↦ pe ∉ content not theorem
    THEN
        act1:   content ≔ content ∪ {rc ↦ pe}
    END
END
```

Problem

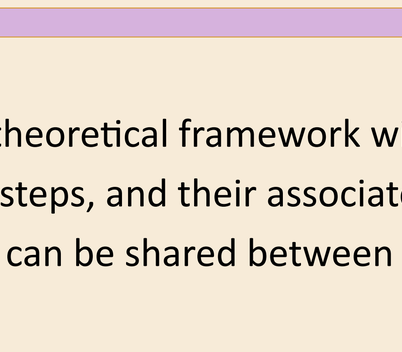
Different formalisms do not integrate well e.g. Event B models the specification it does nothing for the implementation and its proofs are not easily transferable to other formalisms.

Solution

- 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 can provide this framework and, we will construct an institution based specification of the Event B formalism.

Institutions

- Category Theory is a special branch of Mathematics that allows us not only to describe objects but also to investigate the relationships between them.



- Institutions are an application of category theory that allow us to relate the syntactic and semantic structures of different formal languages [Goguen and Burnstall, 1992].

Our Research Questions

RQ1: Can the theory of institutions ensure the accuracy of the translation between Event B and other specification formalisms?

RQ2: Can this theory allow us to investigate proof obligations generated by Event B in different formalisms?

Work completed to Date

A series of Event B case studies have been successfully modelled and verified.

- Social Network
- Celebrity Riddle
- Traffic Lights
- Maximum value in an Array

Predicted Outcomes

- Ultimately this work will lead to a more efficient approach to Model Driven Engineering and hence a framework for improved software development.
- The net effect will be higher quality and more reliable software—a major benefit to every community.

References

Abrial, J.-R., 2010 Modeling in Event-B: System and Software Engineering, New York: Cambridge University Press.

Abrial, J.-R., Butler, M., Hallerstede, S. and Voisin, L., 2006,"An Open Extensible Tool Environment for Event-B," *Lecture Notes in Computer Science, vol. 4260,* pp. 588-605.

Britton, T., Jeng., L,. Carver, G., Cheak, P., and Katzenellenbogen, T., 2013. "Reversible Debugging Software".

Charette, R. N., 2005 "Why Software Fails," *IEEE Spectrum,* pp. 42-49.

Goguen, J. A. and Burstall, R. M., 1992 "Institutions: Abstract Model Theory for Specification and Programming," *Journal of the Association for Computing Machinery, vol. 39,* no. 1, pp. 95-146.

Kumar, C. S., Raghu, D. and Kumar, P. R., 2013 "A Domestic Case Studies Probability to Overcome Software Failures," *Journal of Telematics and Informatics, vol. 1,* no. 1, pp. 20-25.

For more information see: Principles of Programming Research Group http://www.cs.nuim.ie/research/

"The most important property of a program is whether it accomplishes the intention of its user." — C.A.R. Hoare