
University of
Southampton

University of Southampton

Building an Extensible Textual Framework for the Rodin Platform

T.S. Hoang, C. Snook, D. Dghaym, A. Salehi Fathabadi, and M. Butler
ECS, University of Southampton, U.K.

F-IDE 2022 Workshop
26/09/2022, Berlin, Germany

University of Southampton

Outline

- Background
  - Event-B
  - Rodin
  - The need for textual representation
- Design of CamilleX
  - Basic design
  - Direct and indirect extensions
- Summary
  - Future work
  - Lessons learnt and important design decisions

University of Southampton

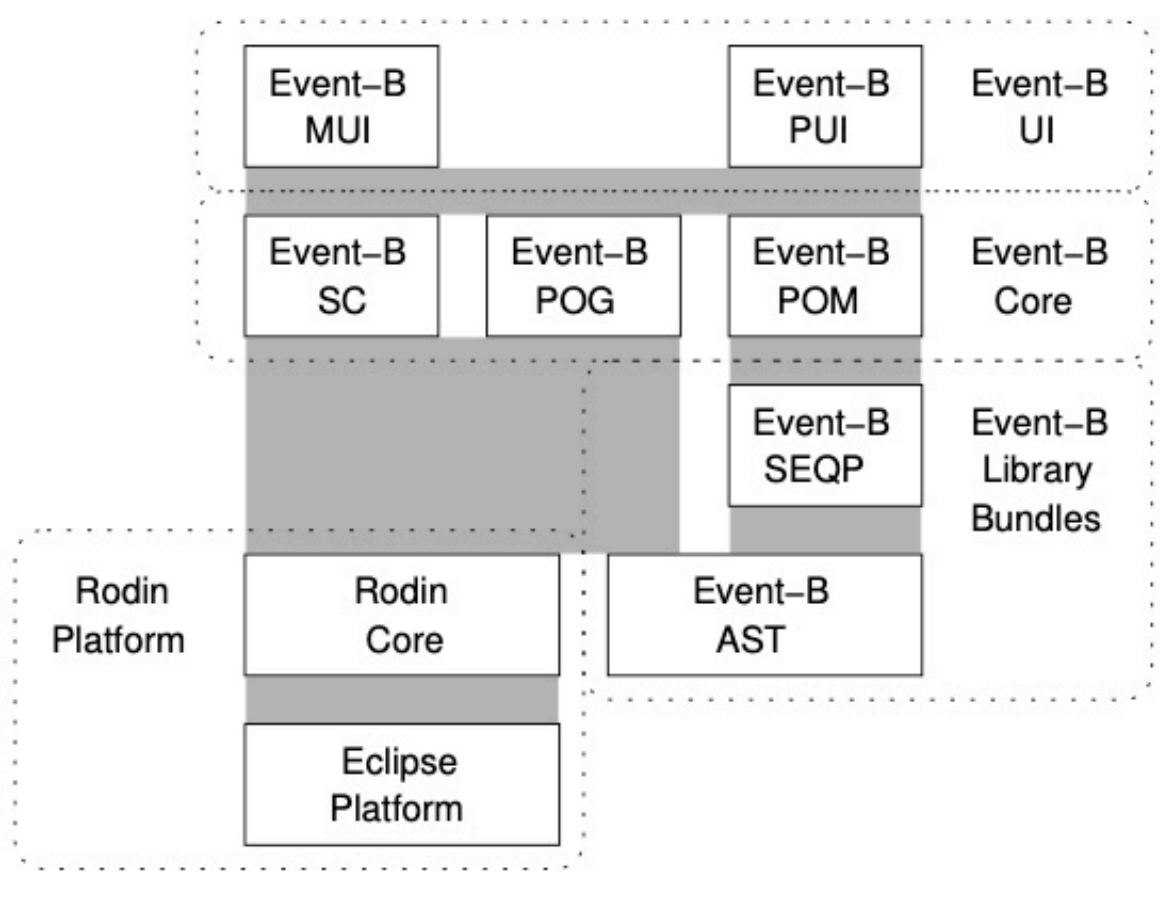
Background

Event-B

- Discrete transition systems
  - Variables representing states
  - Guarded events representing transitions
  - Contexts: Static part of the models (carrier sets, constants, etc.)
  - Machines: Dynamic part of the models (variables, events, etc.)
- First-order logic with set theory
- Highly extensible
  - The theory plug-in
  - UML-B
  - etc.

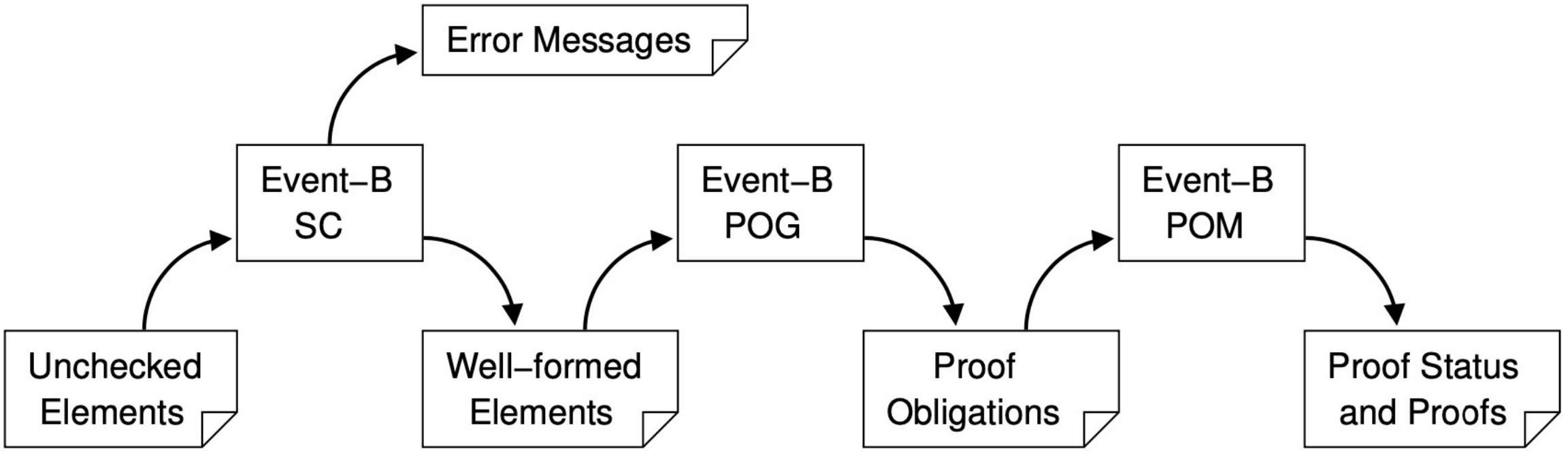
Background

Rodin Platform Architecture



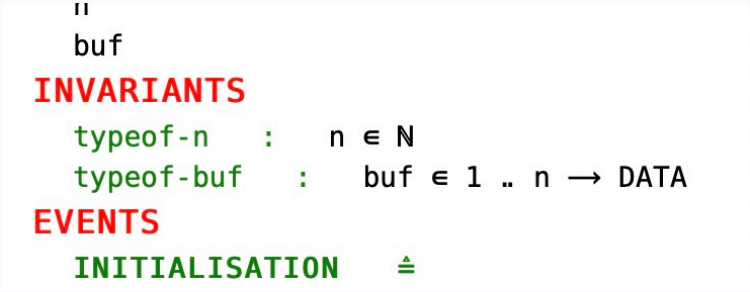
Background

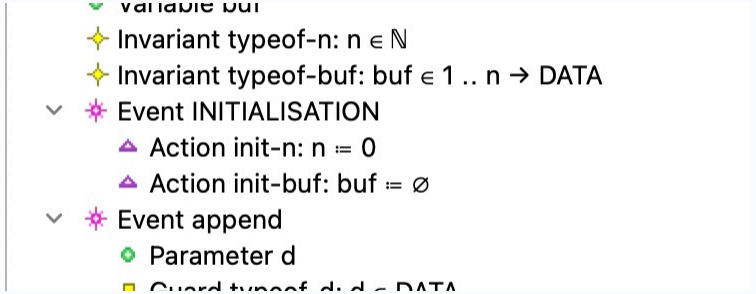
Rodin Platform Builder



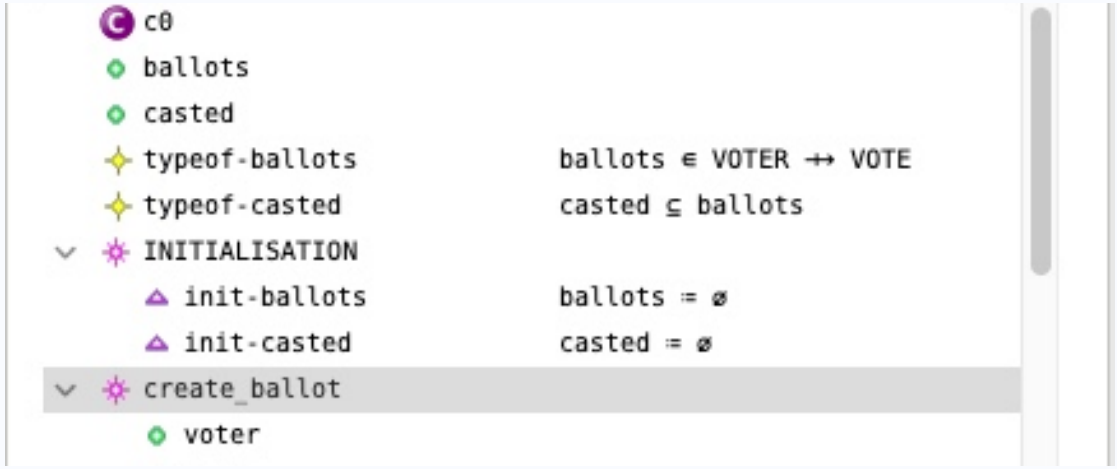
The Challenge

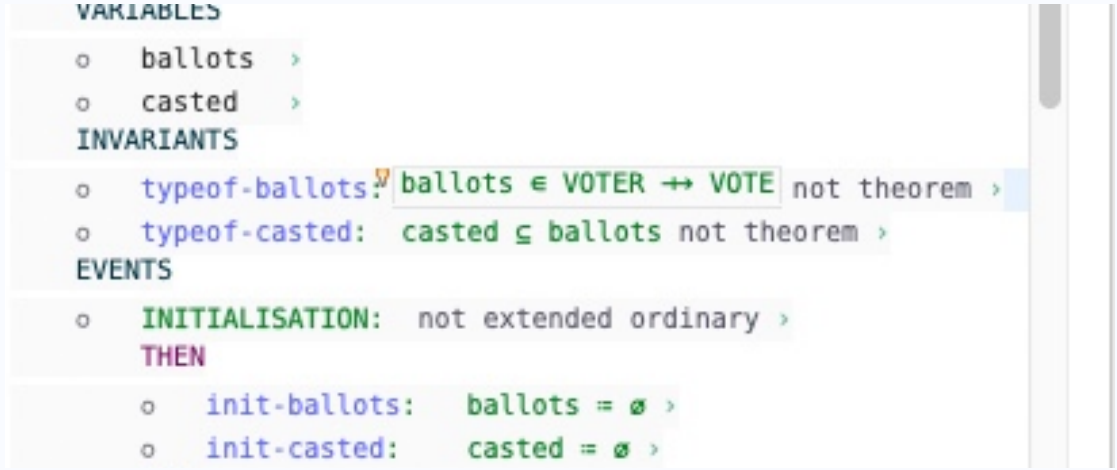
- Event-B models do not have an “outer” syntax
- Models are serialised in XMI format …
- … stored in the Rodin Database (tree-structured).
- Developing a user-friendly editor for Event-B models is challenging

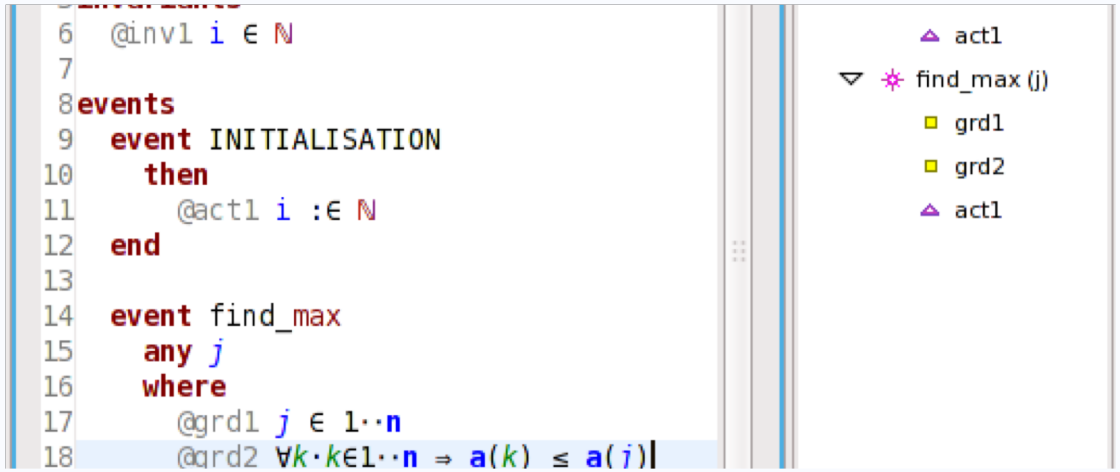




The Challenge – Editors for Tree-Based Structured







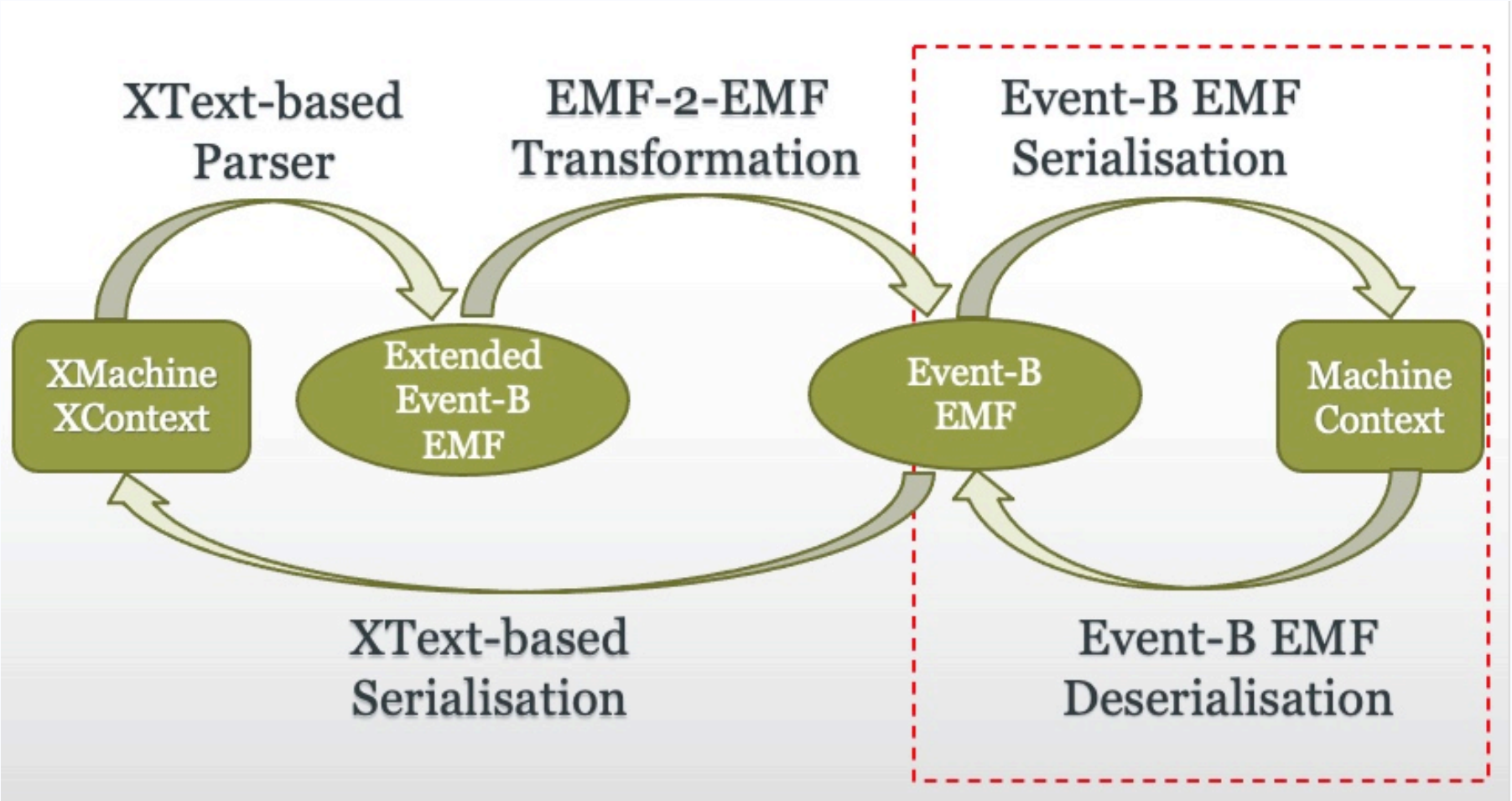
University of Southampton

The Need for Textual Representation

- (True) Textual representation helps with teamworking
- Framework (e.g., XText) for developing IDE for DSLs.
- Design Principles:
  1. Reuse the existing Event-B tools of Rodin as much as possible.
  2. Support direct extension of the Event-B syntax to provide additional features.
  3. Provide compatibility with other kinds of ‘higher-level’ models that contribute to the overall model, e.g., UML-B diagrams.
- We make use of the Event-B EMF and EMF-2-EMF framework

University of Southampton

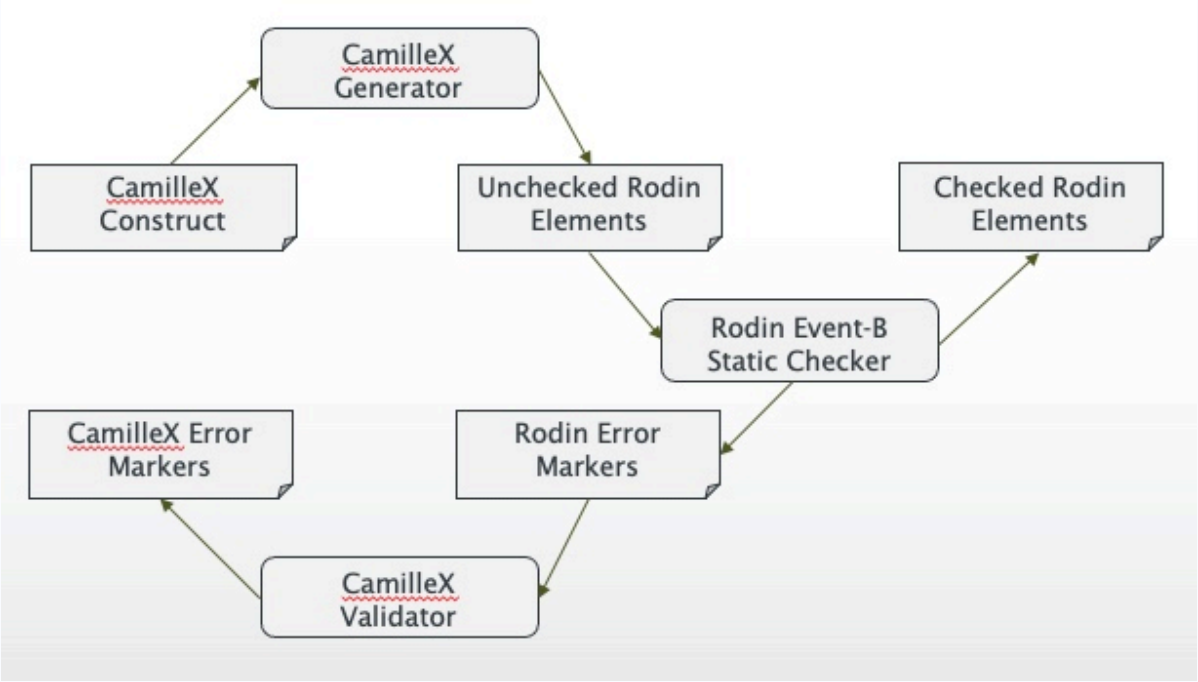
The CamilleX Framework



The CamilleX Framework

Main Ideas

- Provide the "outer" syntax for Event-B models
  - Use Rodin for static checking of "inner" syntax including mathematical formulae
  - Call-back mechanism to report errors/warnings to CamilleX



The CamilleX Framework

Results

- True textual representation
  - Teamworking is fairly easy
  - other useful features: comments everywhere (even within a formula)
- Straight-forward to extend:
  - Direct extensions: Extending the CamilleX grammar
    - Machine inclusion (Hoang et al. [2017]): composition of models
    - Record structure (Salehi Fathabadi et al. [2021])
  - Indirect extensions: via the generic containment mechanism:
    - UML-B etc.

University of Southampton

Steps for Direct Extensions

1. Extend the Event-B EMF with new modelling elements.
2. Extend the grammar of the CamilleX constructs and regenerate the supporting tools.
3. Extend the CamilleX validator to ensure the consistency of the added modelling elements.
4. Extend the CamilleX generator to translate the newly added modelling elements.

University of Southampton

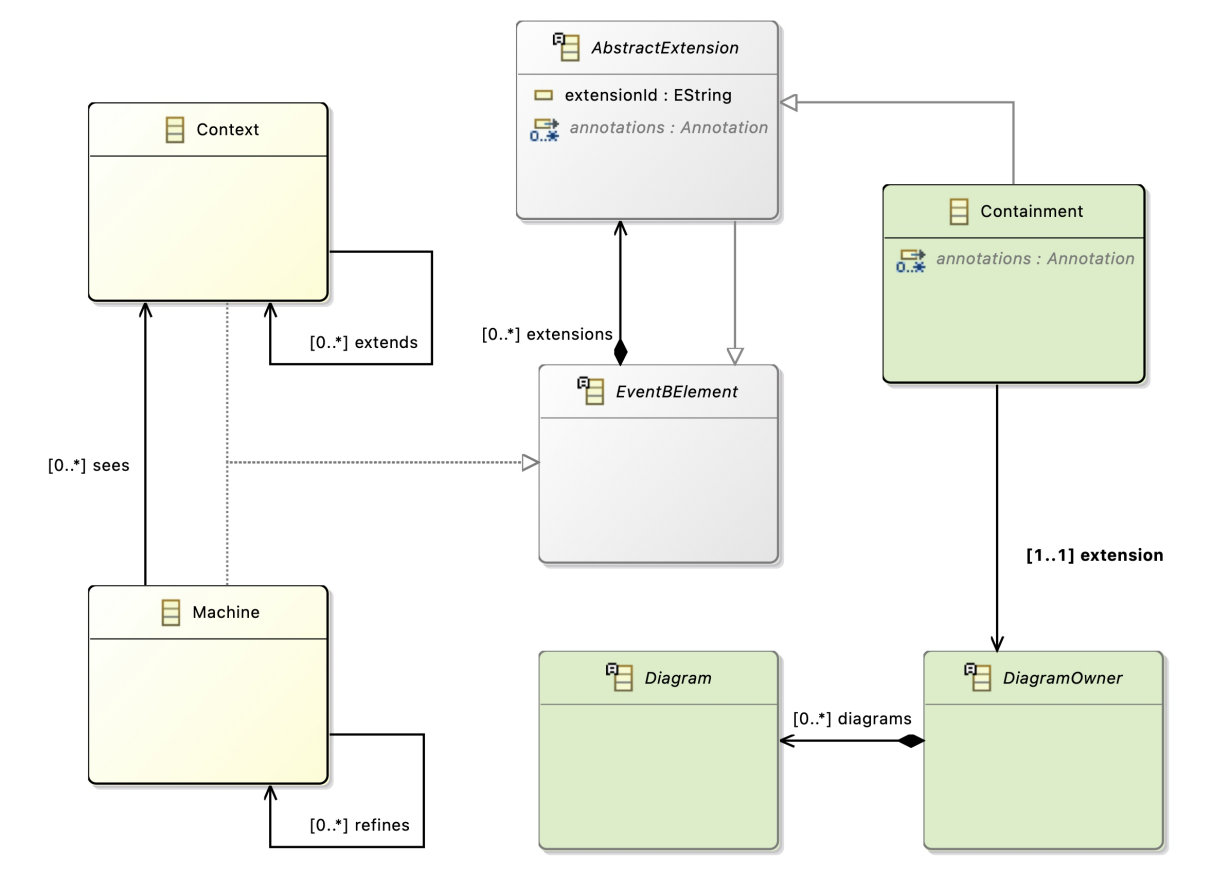
Direct Extensions Demo

- Machine inclusion
- Record structure

Indirect Extensions

Generic Containment Mechanism

- Machines and Contexts can contain zero or more DiagramOwner.
- An extension point is created for the CamilleX generator
  - Plug-in can provide an implementation for translating a specific subclass of DiagramOwner.
- Any sub-class of DiagramOwner can be contained in Machines and Contexts
- The contained model does not need to be serialised using XText.



University of Southampton

Summary

- Textual presentation of Event-B models
  - Highly-extensible
  - Direct syntax extensions
  - Indirect extensions via the generic containment mechanism
- Future Work
  - Machine inclusion: filter unnecessary proof obligations, incorporate refinement chain, integrate with context instantiation
  - Complete support for record structure refinement (prototyped for CamilleX 3.0.0)
  - Integrate with UML-B and develop XUML-B for textual serialisation.
  - Reasoning about liveness properties
  - etc.

University of Southampton

Main Lesson Learnt/Important Design Decisions

- It is essential to have a textual serialisation for Event-B models.
- We design CamilleX with highly extensible (a principle of Event-B/Rodin)

Direct Extensions	Indirect Extensions
Require regeneration of CamilleX	Do not require regeneration of CamilleX
CamilleX depends on the direct extensions	Indirect extensions depend on CamilleX
Integrated syntax with CamilleX	Models are external/independent of CamilleX
CamilleX must be installed together	CamilleX can be installed independently
CamilleX must be maintained together	CamilleX can be maintained independently

University of Southampton

YOUR QUESTIONS