
University of
Southampton

University of Southampton

Behaviour Driven Formal Modelling

Dr. Colin Snook,

Dr Thai Son Hoang, Dr Asieh Salehi, Dr Dana Dghaym, Pr Michael Butler

IVOIRE project workshop, Lugano, Switzerland,

7th June 2022

University of Southampton

Outline

- Introduction
  - Event-B, UML-B, lifecycles, V&V, tools we use
- Behaviour Driven Formal Modelling
  - The modelling process is based on scenarios
  - We need abstraction and refinement of scenarios
  - The Scenario Checker tools
    - And animation more generally
  - Regression testing for models
- Conclusion
  - Some references, Summary

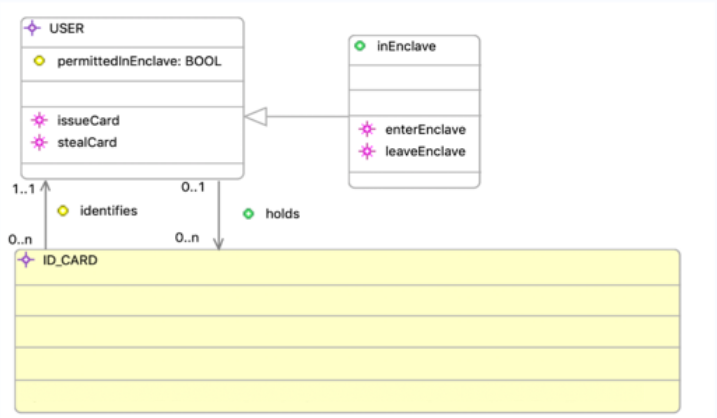
Event-B

- Formal modelling language + tools Systems
  - Sets, predicates, guarded events

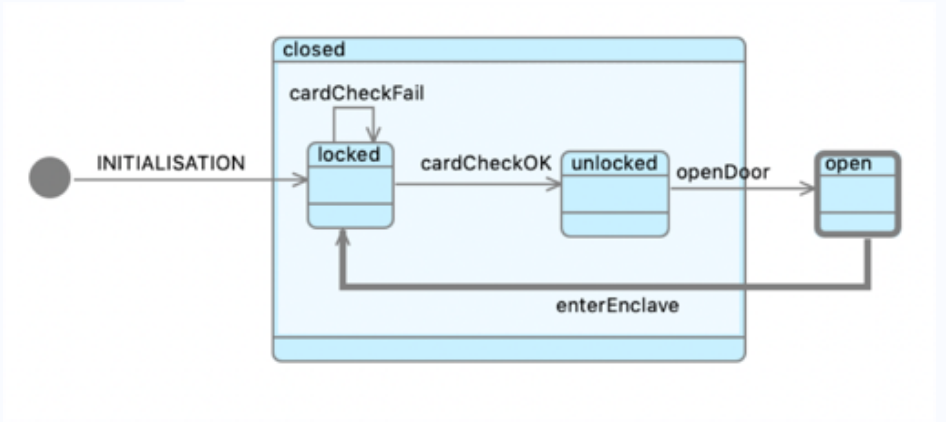
- Refinement
  - Abstract model - Proof of critical properties (invariants)
  - Refine to add detail – proven to preserve abstract properties

UML-B

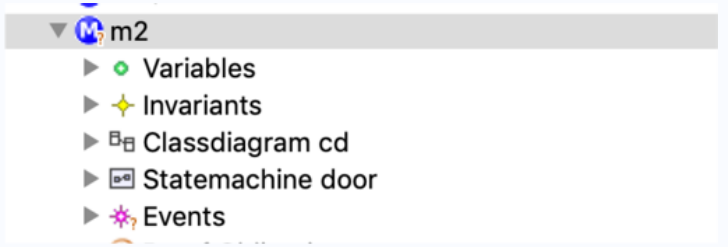
- A formal (mathematical) diagrammatic modelling language
  - Class diagrams, State-machines
  - `UML-like' - but has a different semantics
    - No run to completion (no trigger events)



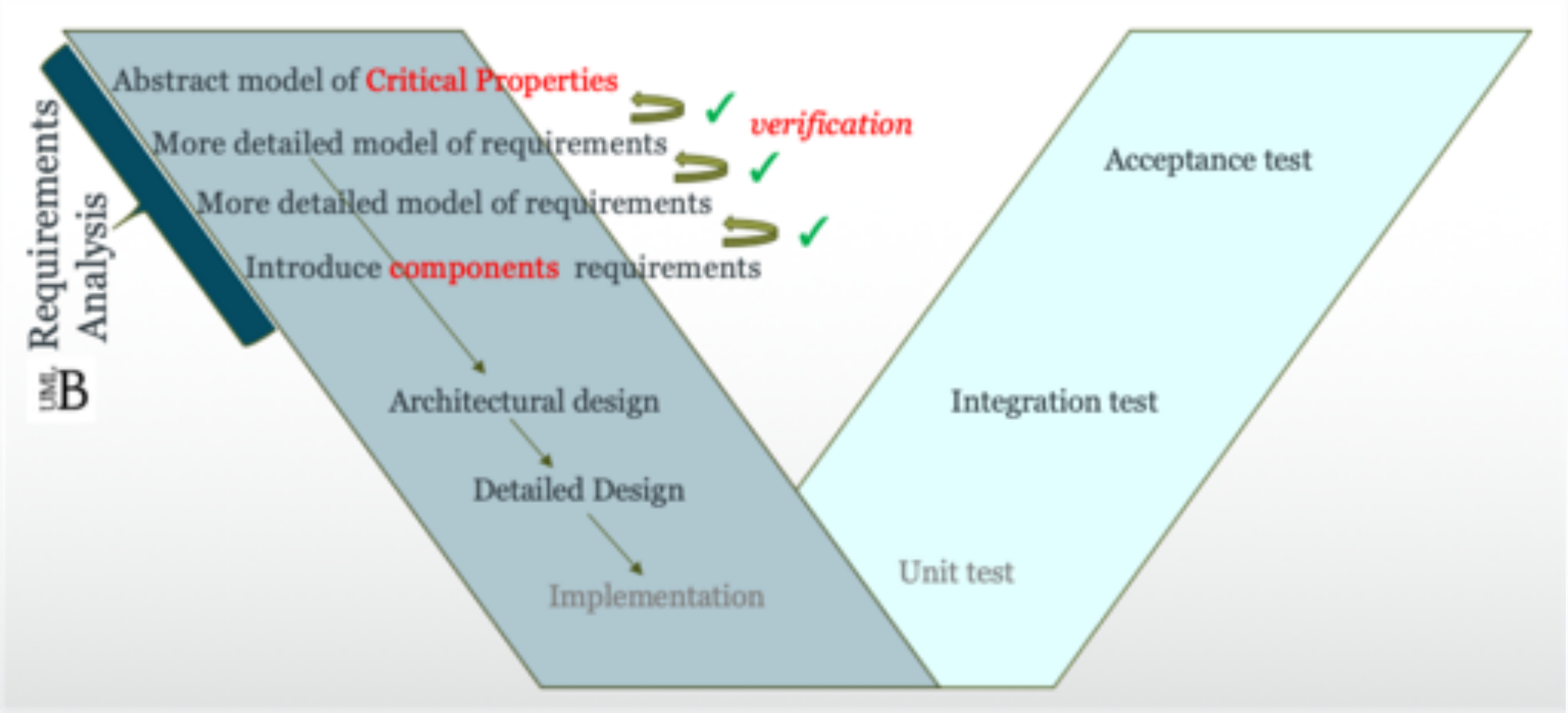
- Refinement
  - Based on Event-B
  - Diagrams generate standard Event-B for verification



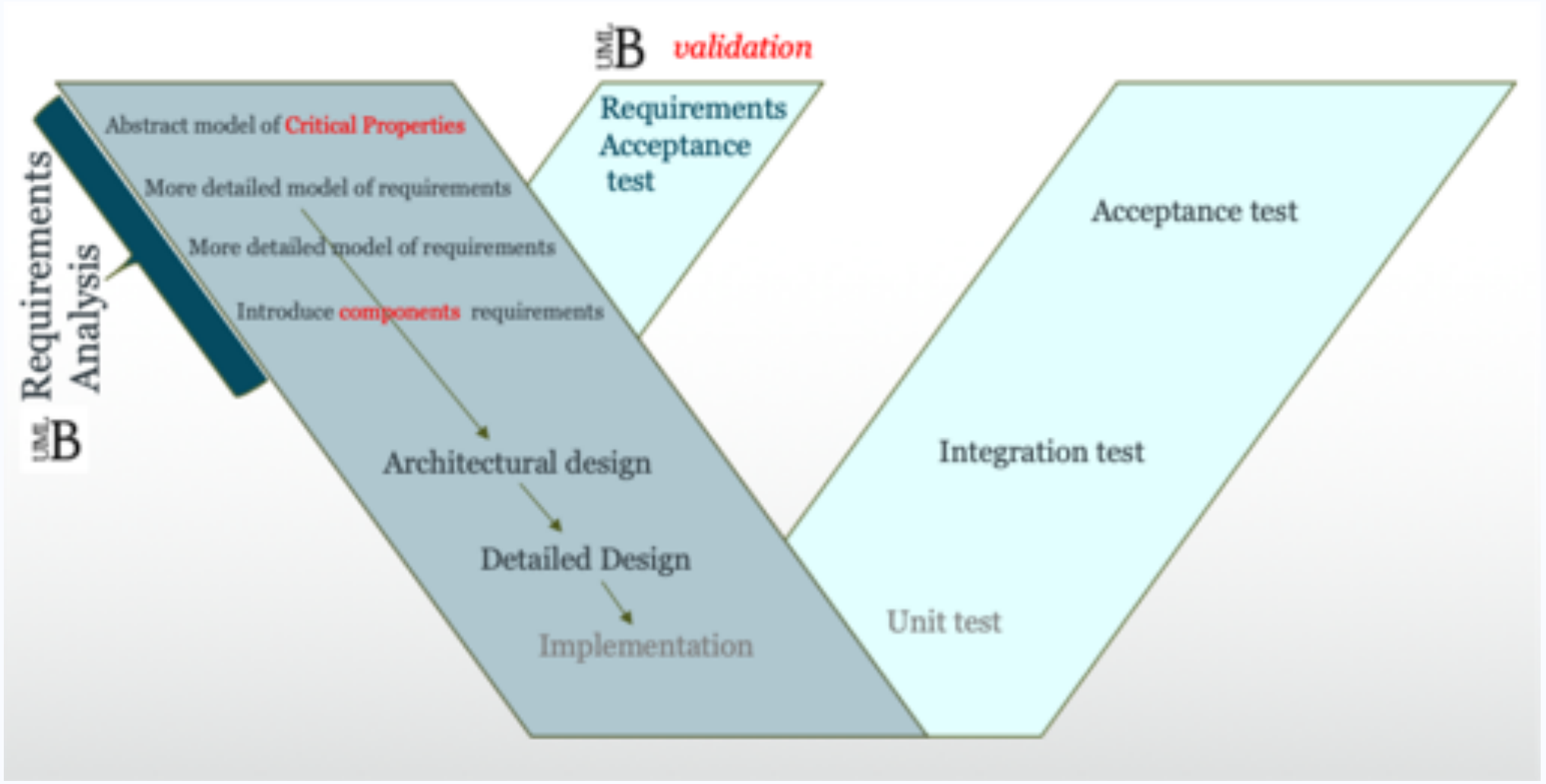
- **State-machine Animation plugin**



Event-B modelling in the lifecycle



Event-B modelling in the lifecycle



Verification to Validation

University of Southampton

- Verification of critical properties by Proof
  - Safety, security properties etc. as invariants
  - NO INSTANTIATION NEEDED
  - If not proven – use model checker to find counter-examples
- Verification of Behaviour
  - Animation + Visualisation
  - Scenarios - illustrate behavioural requirements - contain expected responses
- Model Validation
  - Animation + Visualisation
  - Scenarios still drive animation, but…
  - Domain Experts assess results

Event-B tools

Key:
Systerel and others,
HHU Duesseldorf,
Uni of Southampton,
Clearsy

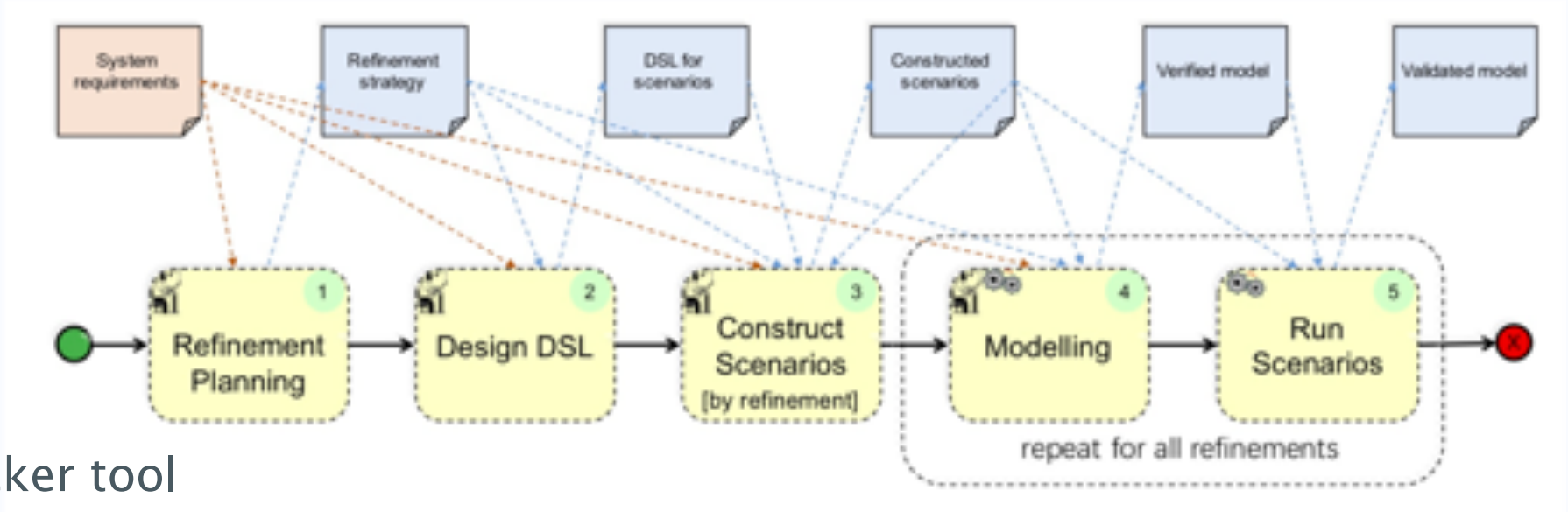
University of Southampton

Type	Tool	Used for
Platform	Rodin	Modelling environment
Modelling	Event-B (Rodin)	Formal modelling language
Modelling	Camille-X	Text-based modelling extensions
Modelling	UML-B	Diagrammatic modelling
Theorem provers	Proof framework (Rodin), AtelierB provers, SMT provers	Critical Properties, Refinement, Well-definedness
Model checker	ProB	De-bugging models, Data verification
Animation	ProB	Behavioural testing, Acceptance testing
Visualisation	BMotionStudio, UML-B animation	
Scenario player	Scenario checker	
Scenario scripts	Cucumber (from BDD)	Regression testing

Behaviour Driven Formal Modelling

University of Southampton

- Scenarios drive modelling
  - Abstract Scenarios *drive* abstract models
  - Executable DSLs for precision



- Scenario Checker tool
  - Record + replay scenarios
  - Check state against previous recording
  - Internal events fire automatically, Private variables are hidden

Refinement of Scenarios

Abstract Scenario → Refined Scenario

```
Given USER1 is not in ENCLAVE
And USER1 is allowed in ENCLAVE
When USER1 enters ENCLAVE
Then USER1 is in ENCLAVE
And USER1 is allowed in ENCLAVE
```

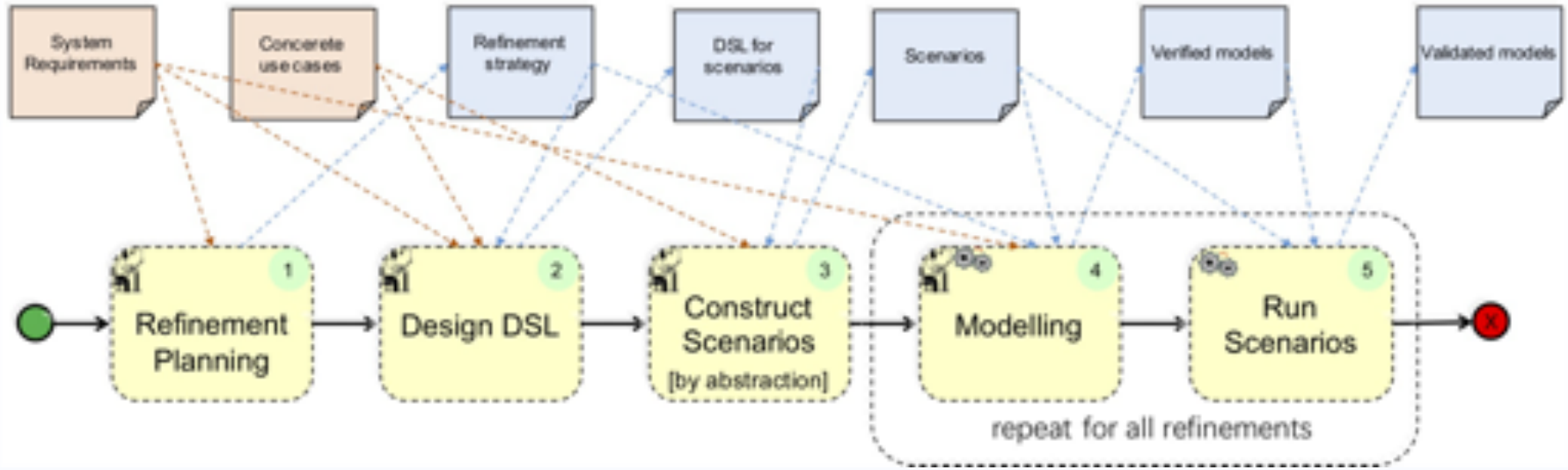
Note that we do not need to re-check abstract states. They are guaranteed by refinement

```
Given USER1 is not in ENCLAVE
And Door is CLOSED
And Latch is LOCKED
And USER1 holds Card1
And Card1 identifies USER1
When USER1 enters APPROACH
Then USER1 holds Card1
And Card1 identifies USER1
When Latch unlocks
When Door opens
When USER1 enters ENCLAVE
When Door closes
When Latch locks
Then USER1 holds Card1
And Card1 identifies USER1
```

Our scenario DSLs are based on the Gherkin language and Cucumber framework used in BDD
https://cucumber.io/docs/gherkin/reference/

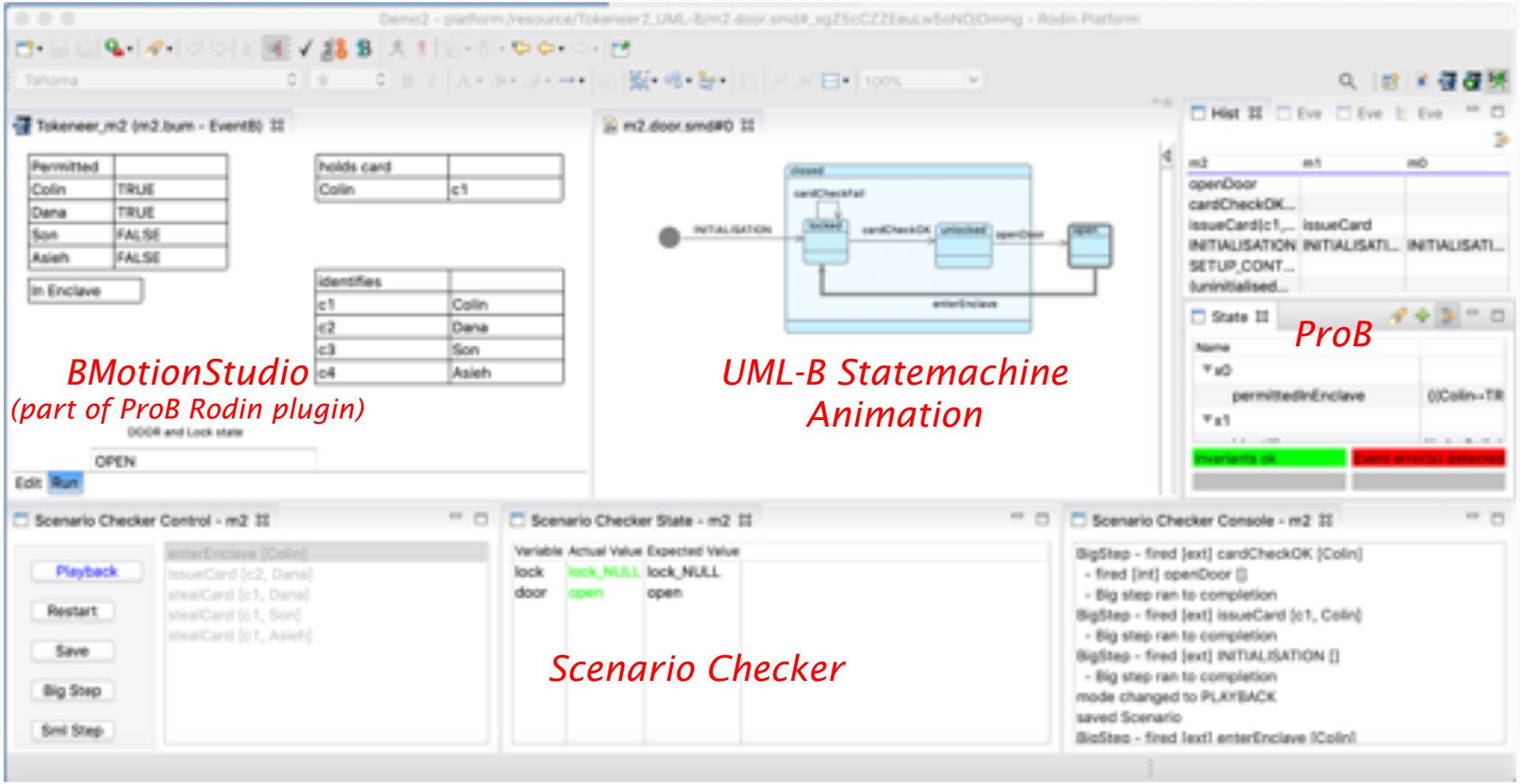
Behaviour Driven Formal Modelling

- Scenarios may be ‘given’
  - They will be natural language, concrete and very detailed



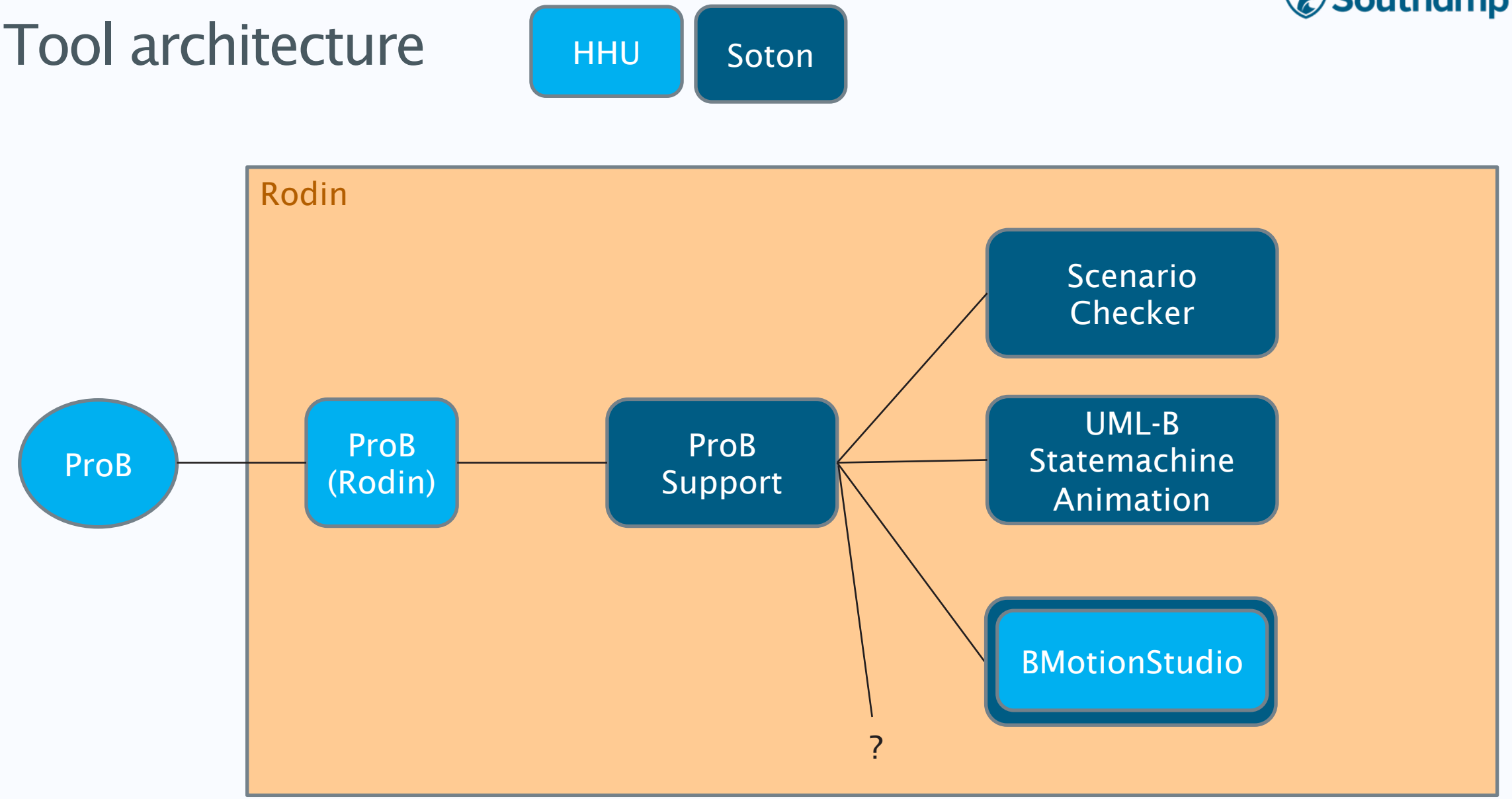
- DSL makes them more precise
- Then construct abstract scenarios by ABSTRACTION

Scenario Checker



Re-playing a pre-recorded scenario

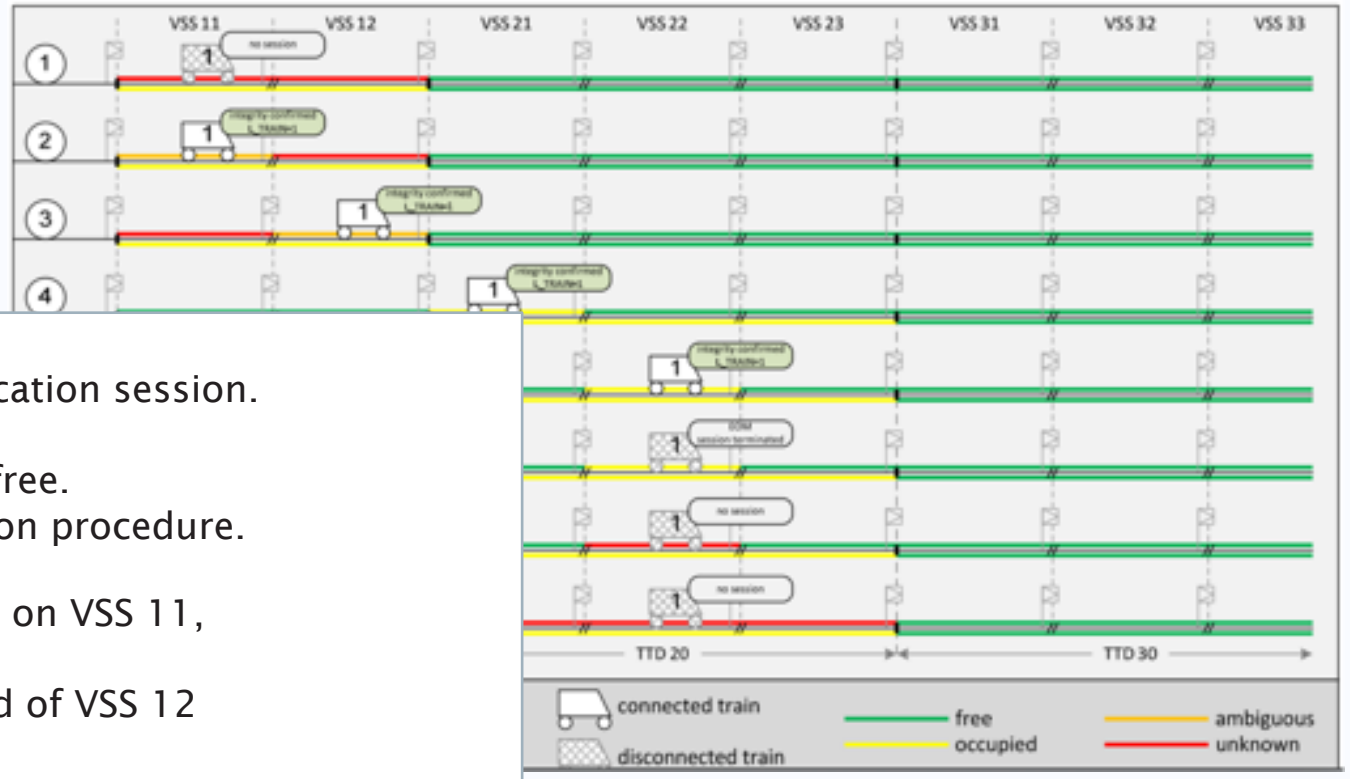
Tool architecture



ETCS/ ERTMS Hybrid Level 3

- European Train Control System
- ERTMS – European Railway Traffic Management System
- Hybrid Level 3 specification – virtual fixed block (but not moving block)
- Train GSM position reporting and integrity monitoring
- Retain existing trackside train detectors (TTD)
- Hybrid – copes with existing trains (using TTD)
- New trains supervised more densely using virtual blocks

Example Scenario from ERTMS HL3 Specification



1. (a) Train 1 is standing on VSS 11
   (b) with desk closed and no communication session.
   (c) All VSS in TTD 10 are unknown.
   (d) TTD 10 is occupied and TTD20 is free.
2. (a) Train 1 performs the Start of Mission procedure.
   (b) Integrity is confirmed.
   (c) Because train 1 reports its position on VSS 11,
   (d) this VSS becomes ambiguous.
3. (a) Train 1 receives an OS MA until end of VSS 12
   (b) and moves to VSS 12
   (c) which becomes ambiguous.
   (d) VSS 11 goes to unknown.
   (e) Train 1 receives an FS MA until end of VSS 22
4. (a) Train 1 moves to VSS 21
   (b) which becomes occupied
   (c) and all VSS in TTD 10 become free, VSS 11 and VSS 12.
   (d) TTD 10 is free and TTD20 is occupied.

Modelling ERTMS Hybrid Level 3

- ERTMS HL3 modelled in UML-B
- Could not prove safe in some scenarios
  - if trains roll backwards,
  - when movement authority revoked
- As a result – ETCS spec revised
- Behavioural verification+validation of model essential
  - B Motion Studio/ProB,
  - driven by UML-B Scenario checker tool

Dghaym, D., Dalvandi, M., Poppleton, M., Snook, C., Formalising the Hybrid ERTMS Level 3 specification in iUML-B and Event-B. *Int J Softw Tools Technol Transfer* **22**

DSL for ERTMS HL3 Scenarios

Nouns

<train> = <label>
<section> = TTDx
<sub-section> = <section>.VSSy
<ma> = <abstract ma> | <concrete ma>
<abstract ma> = MA
<concrete ma> = FSMA | OSMA
<timer> = <sub-section>.DisconnectTimer | <sub-section>. ShadowTimer | <sub-section>.GhostTimer
<section state> = FREE | OCCUPIED
<sub-section sate> = <abstract sub-section state> | <concrete sub-section state>
<abstract sub-section state> = AVAILABLE | UNAVAILABLE
<concrete sub-section state> = FREE | OCCUPIED | AMBIGUOUS | UNKNOWN

Adjectives

<train> stood at <sub-section>
<train> connected | disconnected
<train> in mission | no mission
<train> is integral | is split
<train> has <ma>
<section> is <section state>
<sub-section> is <sub-section state>
<ma> until <sub-section>

Verbs

<train> enters | leaves <sub-section>
<train> connects | disconnects
<train> starts mission | ends mission
<train> splits | couples
<train> receives <ma>
<timer> starts
<timer> expires
<train> reports position
<train> reports position as integral
<train> reports position as split

DSL for ERTMS HL3 Scenarios

Note that the DSL provides for our refinements

Nouns

<train> = <label>
<section> = TTDx
<sub-section> = <section>.VSSy
<ma> = <abstract ma> | <concrete ma>
<abstract ma> = MA
<concrete ma> = FSMA | OSMA
<timer> = <sub-section>.DisconnectTimer | <sub-section>. ShadowTimer | <sub-section>.GhostTimer
<section state> = FREE | OCCUPIED
<sub-section sate> = <abstract sub-section state> | <concrete sub-section state>
<abstract sub-section state> = AVAILABLE | UNAVAILABLE
<concrete sub-section state> = FREE | OCCUPIED | AMBIGUOUS | UNKNOWN

Refinement of ma

Refinement of VSS state

Adjectives

<train> stood at <sub-section>
<train> connected | disconnected
<train> in mission | no mission
<train> is integral | is split
<train> has <ma>
<section> is <section state>
<sub-section> is <sub-section state>
<ma> until <sub-section>

Verbs

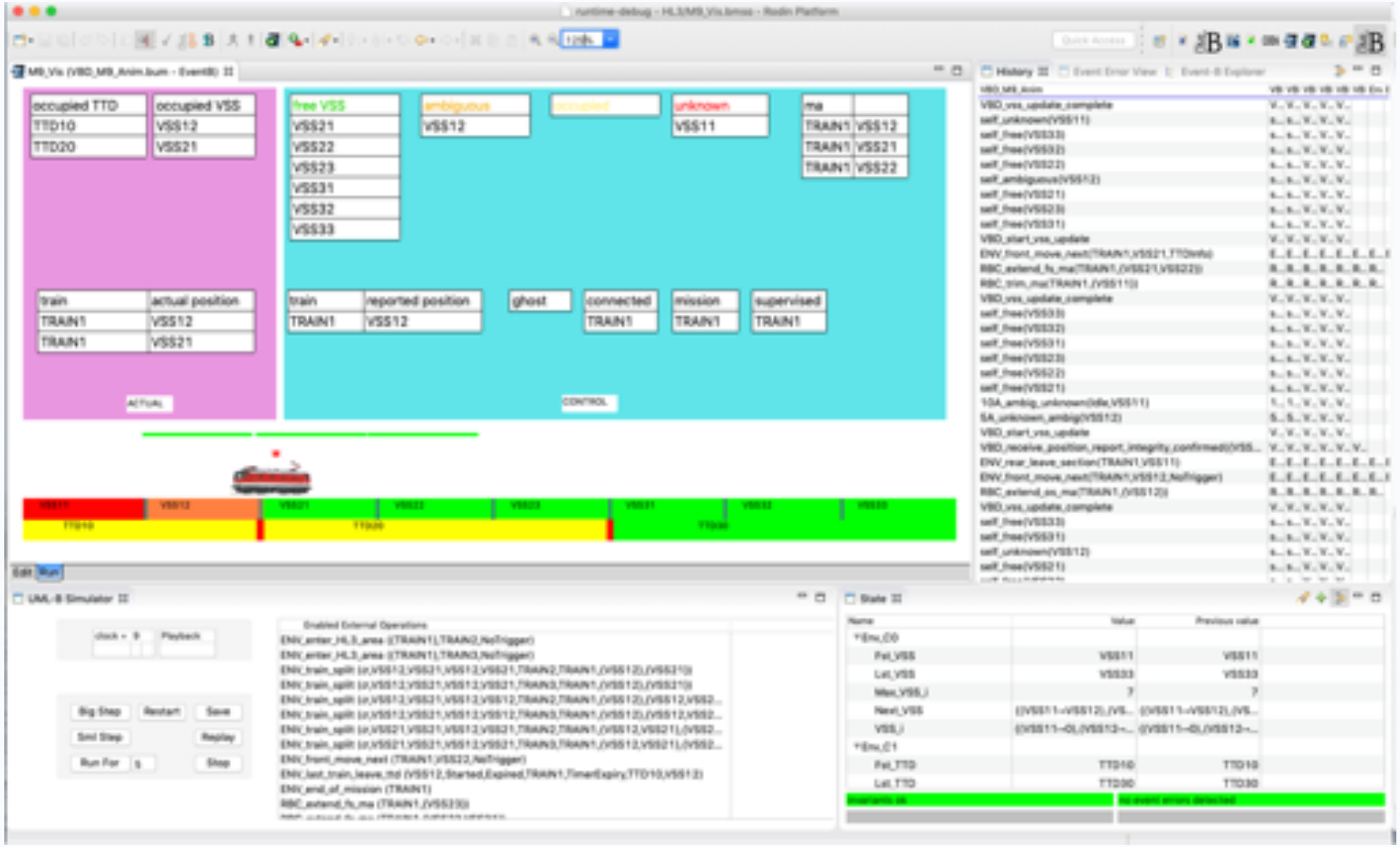
<train> enters | leaves <sub-section>
<train> connects | disconnects
<train> starts mission | ends mission
<train> splits | couples
<train> receives <ma>
<timer> starts
<timer> expires
<train> reports position
<train> reports position as integral
<train> reports position as split

Example Scenario from ERTMS HL3

1. (a) Train 1 is standing on VSS 11
   (b) with desk closed and no communication session.
   (c) All VSS in TTD 10 are "unknown".
   (d) *TTD 10 is occupied and TTD20 is free.*
2. (a) Train 1 performs the Start of Mission procedure.
   (b) Integrity is confirmed.
   (c) Because train 1 reports its position on VSS 11,
   (d) this VSS becomes "ambiguous".
3. (a) Train 1 receives an OS MA until end of VSS 12
   (b) and moves to VSS 12
   (c) which becomes "ambiguous".
   (d) VSS 11 goes to "unknown".
   (e) Train 1 receives an FS MA until end of VSS 22

```
1  Given Train1 stood at TTD10.VSS11
2  And Train1 disconnected
3  And TTD10.VSS11 is UNKNOWN
4  And TTD10.VSS12 is UNKNOWN
5  And TTD10 is OCCUPIED
6  And TTD20 is FREE
7  When Train1 starts mission and Train1
       connects
8  When Train1 reports position as integral
9  Then TTD10.VSS11 is AMBIGUOUS
10 When Train1 receives OSMA until TTD10.VSS12
11 When Train1 enters TTD10.VSS12
12 When Train1 leaves TTD10.VSS11
13 When Train1 reports position as integral
14 Then TTD10.VSS12 is AMBIGUOUS
15 And TTD10.VSS11 is UNKNOWN
16
```

Modelling ETCS Hybrid Level 3



Regression testing using Cucumber

```
When Train1 leaves TTD10.VSS11
```

Gherkin Command in DSL

```
When(~/^${id} leaves ${id}$/) {
 String train, String vss ->
 fireEvent("ENV_rear_leave_section",
    "tr = " + train + " & " + "vss = " + vss)
}
```

Cucumber step definition

```
event ENV_rear_leave_section
any
 tr // The train
 vss // The VSS from that the train moves
where ... then ... end
```

Event-B event

References

- The ERTMS HL3 Specification
  - http://www.ertms.be/sites/default/files/2018-03/16E0421A_HL3.pdf
- Our paper about modelling ERTMS HL3
  - https://link.springer.com/article/10.1007/s10009-019-00548-w
  - Dghaym, D., Dalvandi, M., Poppleton, M., Snook, C., Formalising the Hybrid ERTMS Level 3 specification in iUML-B and Event-B. *Int J Softw Tools Technol Transfer* **22,** 297–313 (2020). https://doi.org/10.1007/s10009-019-00548-w
- Our paper about using Scenarios for formal modelling of HL3
  - https://www.sciencedirect.com/science/article/abs/pii/S1383762120301259
  - Snook, C., Hoang, T.S., Dghaym, D., Fathabadi, A.S., Butler, M. Domain-specific scenarios for refinement-based methods., *Journal of Systems Architecture*, Vol 112 (2021) 101833. https://doi.org/10.1016/j.sysarc.2020.101833

University of Southampton

Summary

- Abstract model of critical properties
  - verified refinements add detail while preserving critical properties
- Verification : Theorem prover + Model checker
  - Gives a deeper understanding of the system
- Validation : Scenario checker - record/replay
  - Acceptance test of requirements **before** implementation
  - Scenario refinement (or it could be scenario abstraction)
  - Scenarios drive the modelling… Behaviour Driven Formal modelling
  - Regression testing of models using scenarios

University of Southampton

Questions?

Visit UML-B.org for more info: