25th anniversary annual INCOSE international symposium
Seattle, WA
July 13 - 16, 2015

Model-based Engineering of Emergence in a Collaborative SoS: Exploiting SysML & Formalism

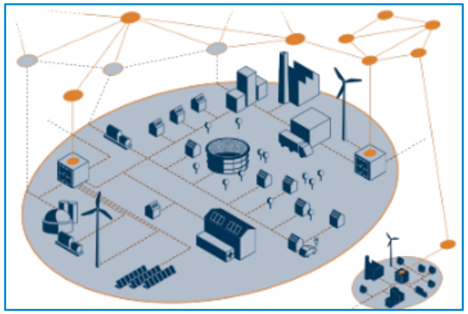
Claire Ingram, Richard Payne, John Fitzgerald
Newcastle University, UK
Luis Diogo Couto, Aarhus University, Denmark

Outline

1. Introduction
   - SoSs, and the challenge of emergence
   - Can formal model-based methods assist?
2. An Integrated Engineering Approach
3. A Pilot Study
   - Requirements modelling
   - Architectural modelling
   - Transition to formal modelling
4. Conclusions

Introduction

- Systems of Systems (SoSs) are comprised of elements that are themselves independent systems
- Often exhibit:
  – Operational & managerial independence
  – Distribution
  – Emergence
  – Evolution
- Types: directed, acknowledged, collaborative, virtual
  – We concentrate on *collaborative* SoSs, which lack a central engineering authority



Emergent Behaviour

- Global behaviour resulting from the interaction of constituent systems
- In SoS, reliance may come to be placed on some emergent behaviour
  - Need to generate evidence confirming/refuting emergent properties
- Engineering for emergence in SoS is challenging
  - Particularly in collaborative SoSs, lacking central decision-making authority
  - Development of techniques to engineer SoS emergent behaviour identified as a key challenge

Can model-based and formal techniques help?

- Model-based techniques can be helpful for SoSs
  - Testing in a realistic environment is difficult or prohibitively expensive
  - Many SoSs required to deliver dependable behaviour in challenging environments
- COMPASS developed model-based SoS Engineering methods
  - Focus on composition: contractual <assumption, commitment> description styles.
  - Focus on semantics (common meanings and ontology)

COMPASS

thecompassclub.org

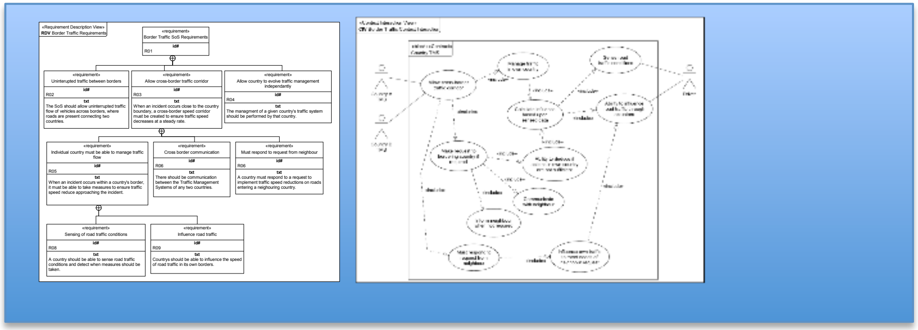
Can model-based and formal techniques help?

- Formal modelling languages have a rigorous mathematical semantics
  - Employed to develop unambiguous models of software-intensive systems
  - Like mathematical models in other engineering disciplines, can be used to generate predictions about the finished system and its behaviour
  - Permit machine-assisted rigorous analysis and verification of requirements and design choices

Can model-based and formal techniques help?

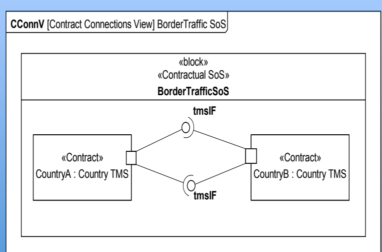
- We blend techniques from systems engineering and software engineering into an approach for analyzing emergent behaviours
  – SysML modelling employed for reasoning about architecture of SoS
  – "Formal methods" already used for verification of dependable software systems are here adapted to validate aspects of SoS behaviour

An Integrated Approach



Requirements Engineering

- **SysML** using a disciplined approach (SoS-ACRE)



«block»
«Operation»
determineSpeedCorridor

parameters
(startLoc:int, distance:nat)

postcondition
forall a in set elems RESULT @
(actMap(a).loc >= startLoc and actMap(a).loc <= distance+startLoc) and
RESULT not in set currCorridors

precondition
len actSeq > 1

return
RESULT:Corridor

Architectural Modelling

- **SysML & CML annotations** using defined patterns

```
process CountryTMS =
  actions
  BEHAVIOUR= NEW_INCIDENT []  NEIGHBOUR_REQ

  NEW_INCIDENT = inIncident.myId?l?t ->
        (dcl d : nat := calcDistance
          (t, nationalSpeedLimit) @
                  CORRIDOR(l, t, d))

  COR
```

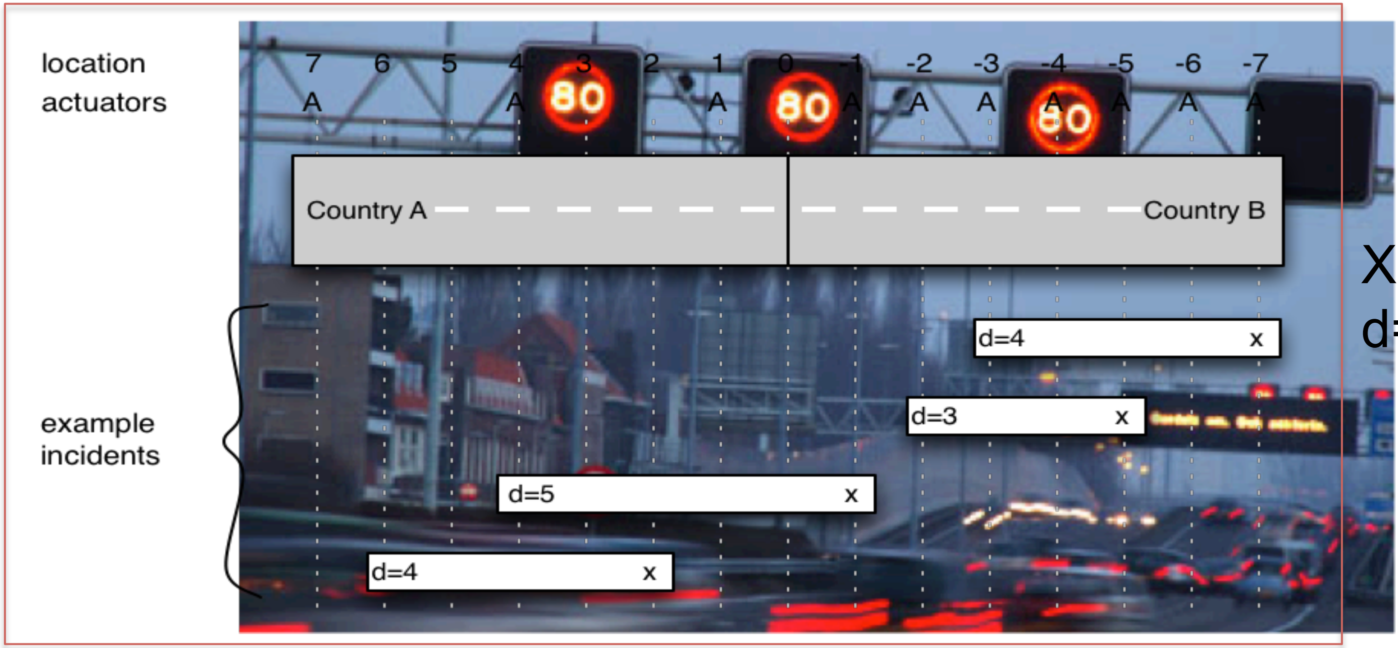
```
process CountryA = CountryTMS(theAId,
                theBId, limitA, actCorrA)
process CountryB = CountryTMS(theBId,
                theAId, limitB, actCorrB)

process BorderTrafficSoS = CountryA [|interface|] CountryB
```

Formal V&V of Models

- **CML** using dedicated tools exploiting the formal semantics

25th anniversary
annual INCOSE
international symposium
Seattle, WA
July 13 - 16, 2015

Example



X = incident
d= length of corridor

- Traffic driving at high speed should not encounter stationary or slow vehicles suddenly; temporary “corridor” is created to ensure approaching vehicles decelerate gradually
- If accident happens near an international border, the corridor may straddle the border & require two countries to co-operate
- Need to define contract to which each country adheres

Requirements Engineering

- COMPASS SoS-ACRE requirements process provides a structured way of engineering and managing the requirements of an SoS
  - Example focuses on the requirements engineering process

- SoS-ACRE requirements engineering steps:
  - Identify source elements of requirements
  - Identify the constituents and stakeholders of the SoS
  - Define the SoS requirements
  - Examine the SoS requirements in context
  - Identify scenarios for validating the requirements

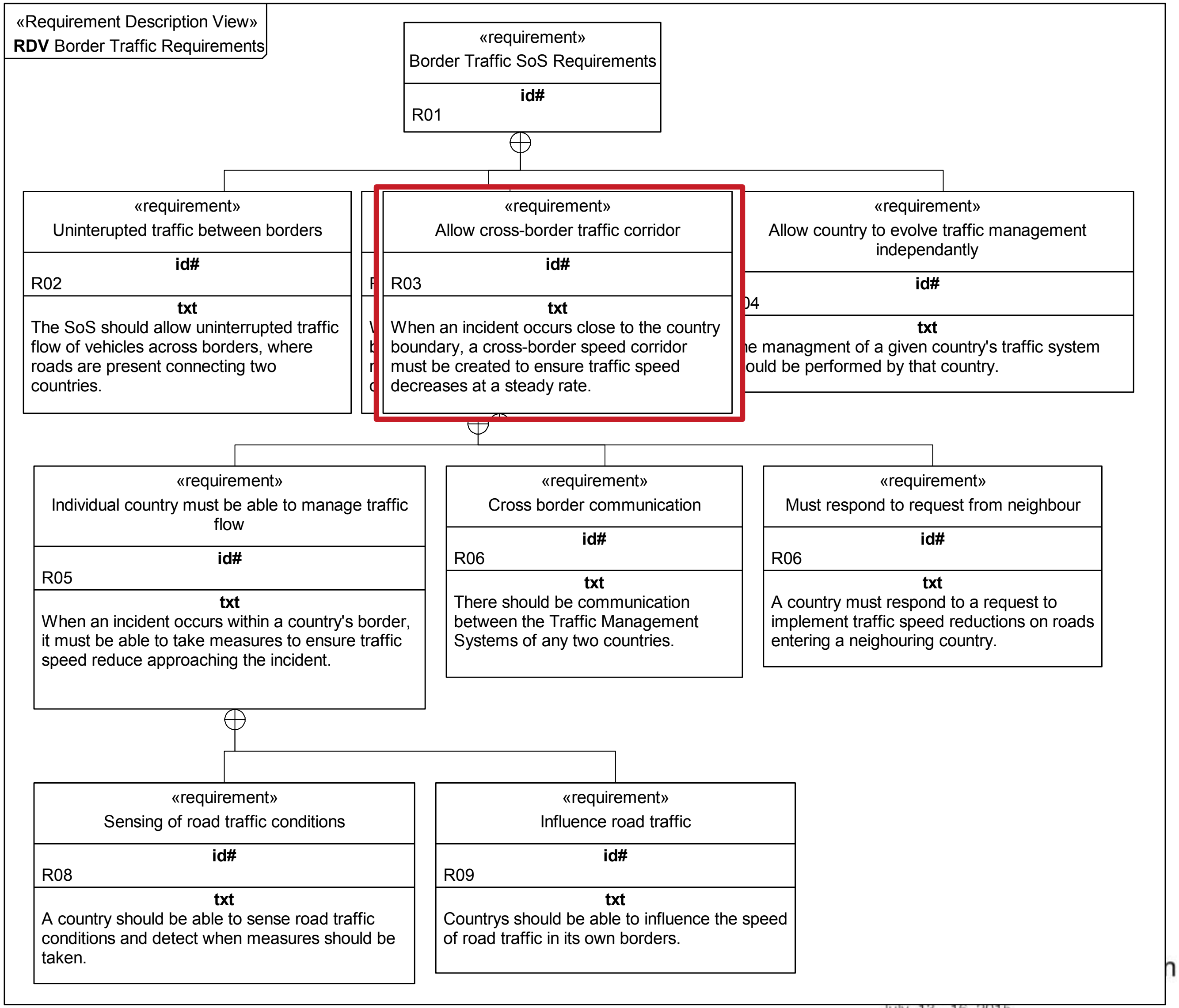
SoS-ACRE Viewpoints

INCOSE 2015

Viewpoint	Description
Source Element	Identifies requirements source information
Requirement Description	Contains a structured description of each requirement
Context Definition	Identifies the points of view (contexts) which will be explored in the RCVs
Requirement Context	Describes the requirements defined on the RDV in context
Context Interaction	Overview of relationships between the contexts of various CSs; combines the RCVs for each CS
Validation	Provides the basis for demonstrating that requirements can be validated
Validation Interaction	A combined view of the scenarios for related use cases in the SoS; combines the VVs of the use cases

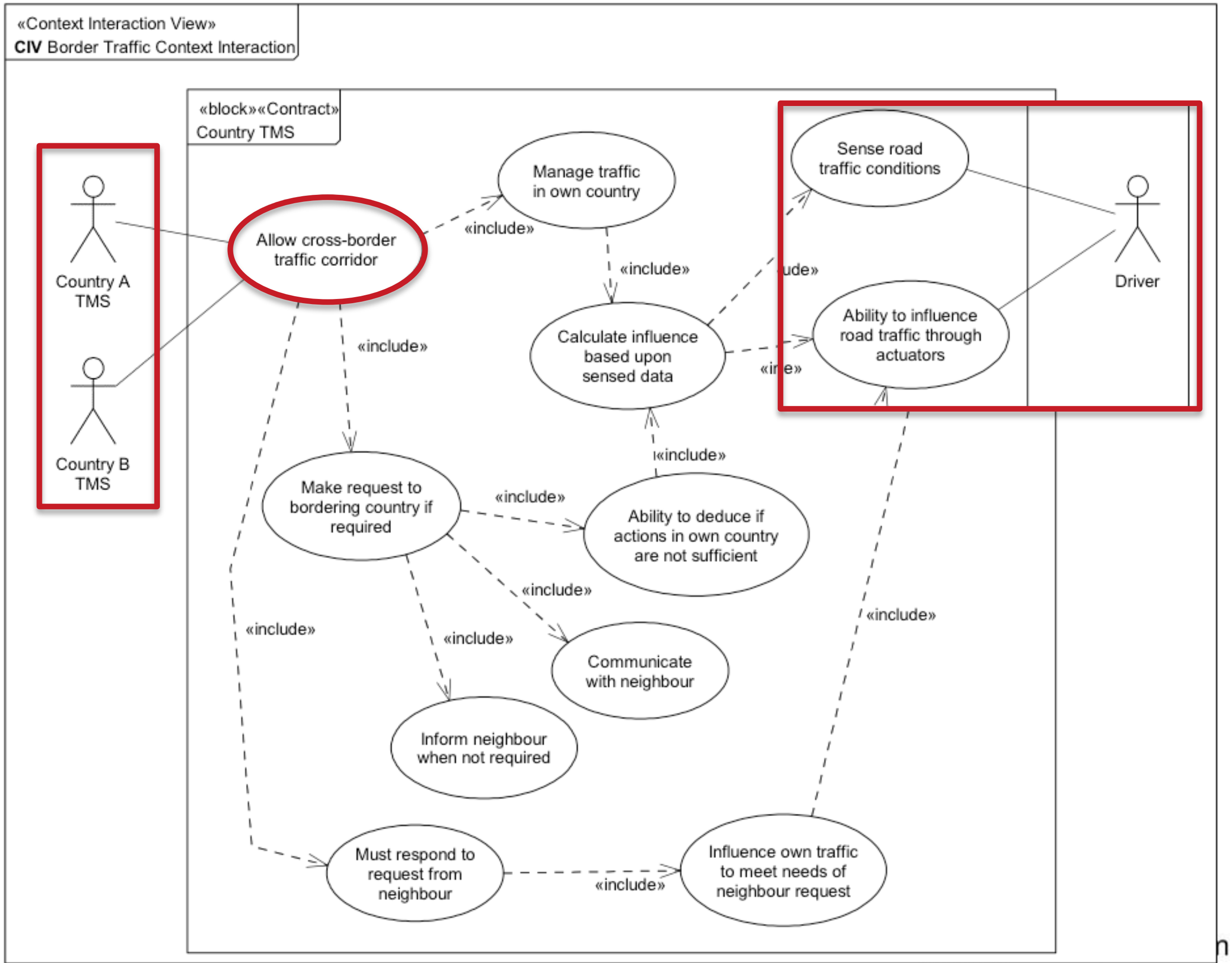
Requirement Definition

- Requirements Description View
- We will focus on one requirement



Requirements in Context

- This view draws different contexts together as (here) use case diagrams for different stakeholders
- Identify duplicates & conflicts
- Some stakeholders may be internal e.g. countries A & B; some are external e.g., driver

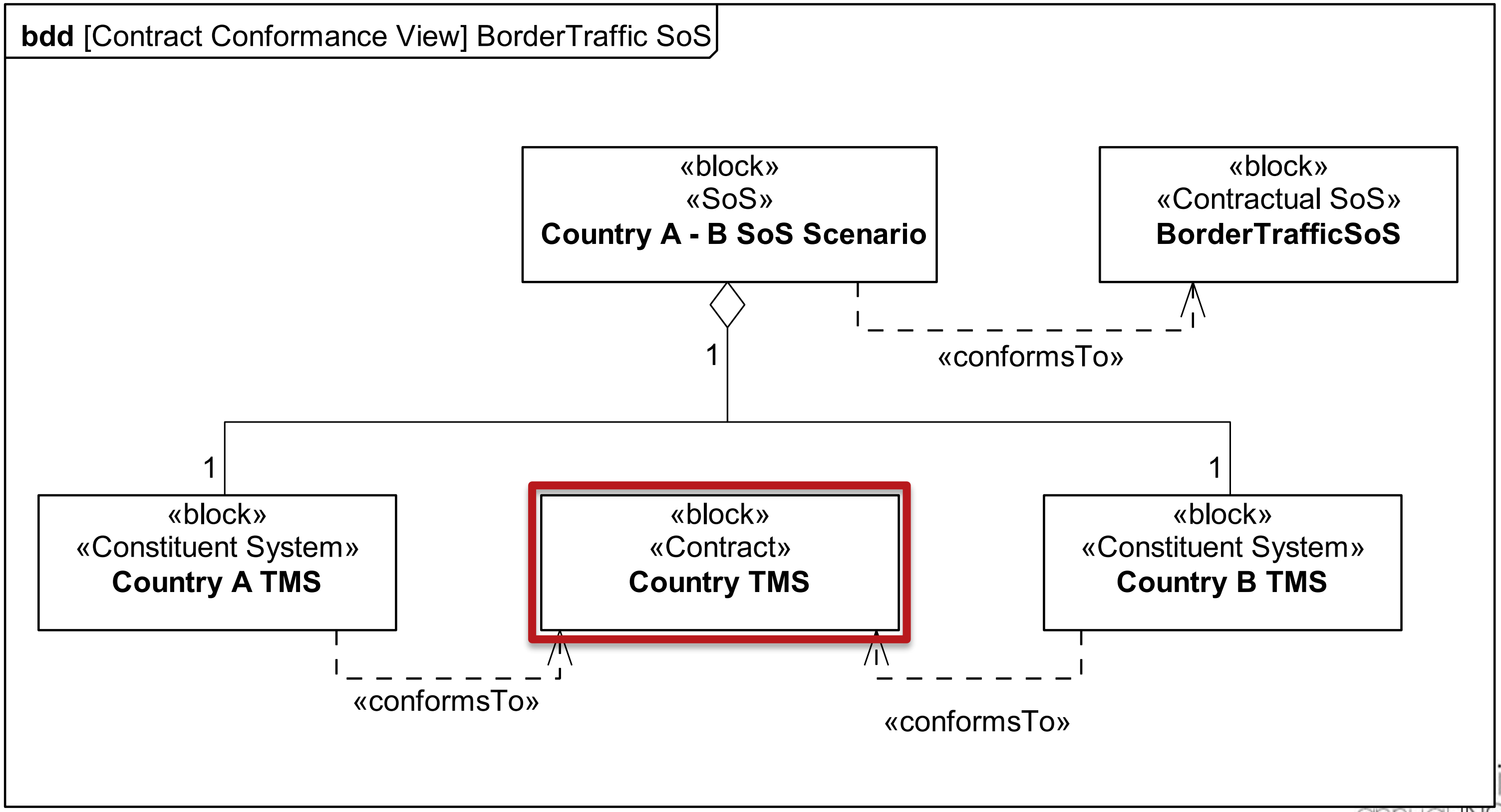


Architectural Modelling

- When defining a SoS architecture, follow COMPASS architectural approach
  - patterns and guidelines
- Use collections of *modelling patterns* to define SoS structure and behaviour
- COMPASS architectural modelling approach also includes *guidelines* for SoS integration and development lifecycles
- In border traffic example, we define the behaviour required by each country's TMS – using the *interface contract* pattern (shown over next few slides)

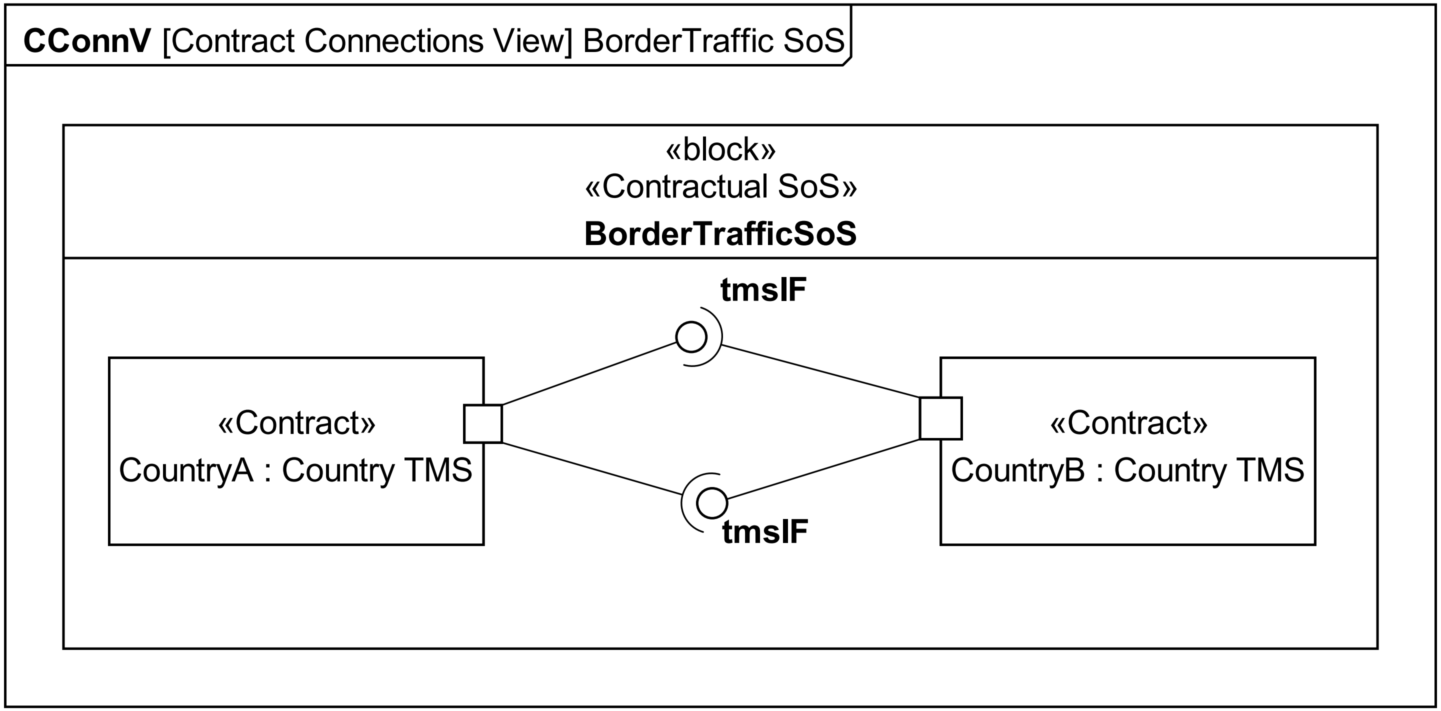
Architectural Modelling

- Identifying contract conformance



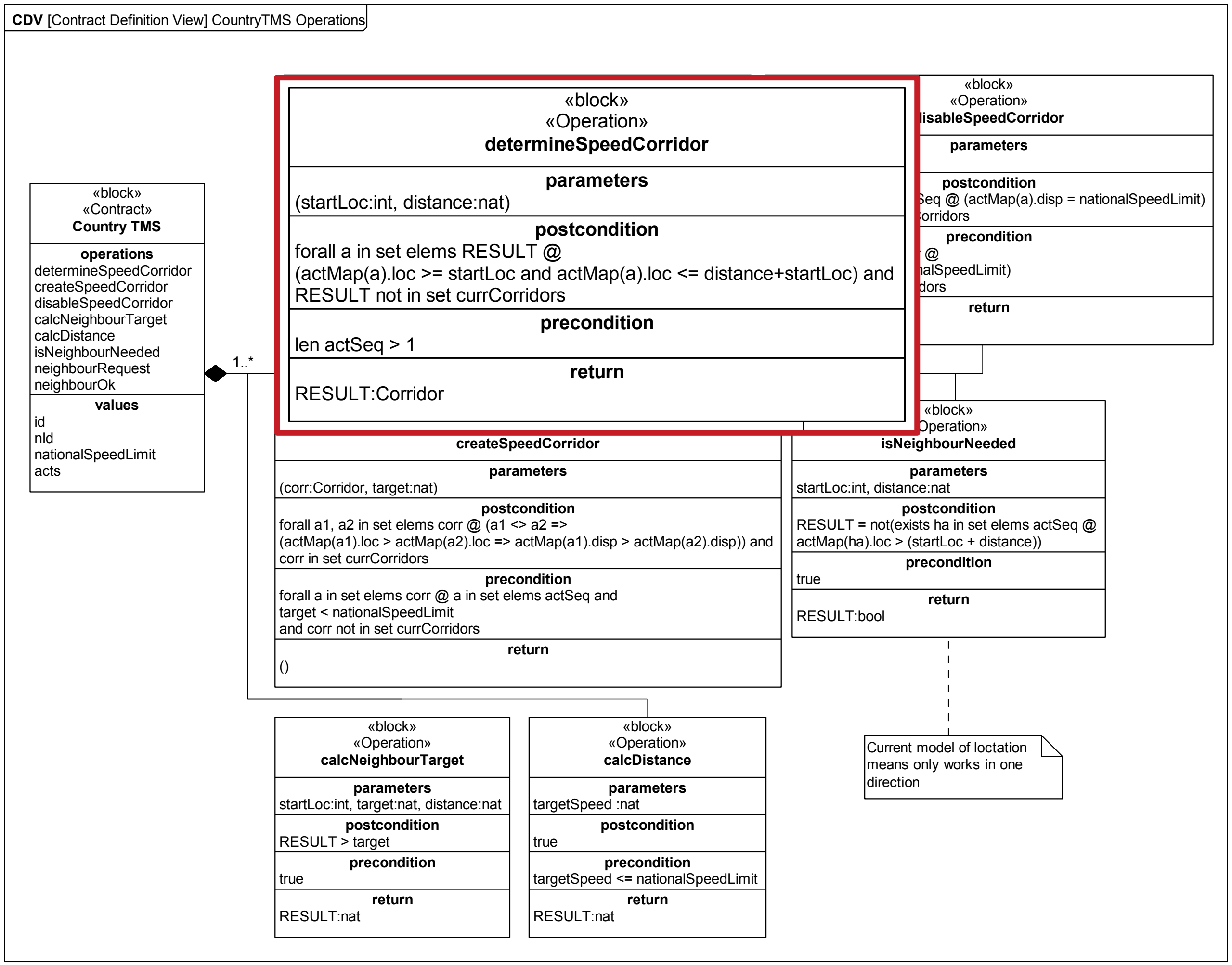
Architectural Modelling

- Defining connections and interfaces between systems



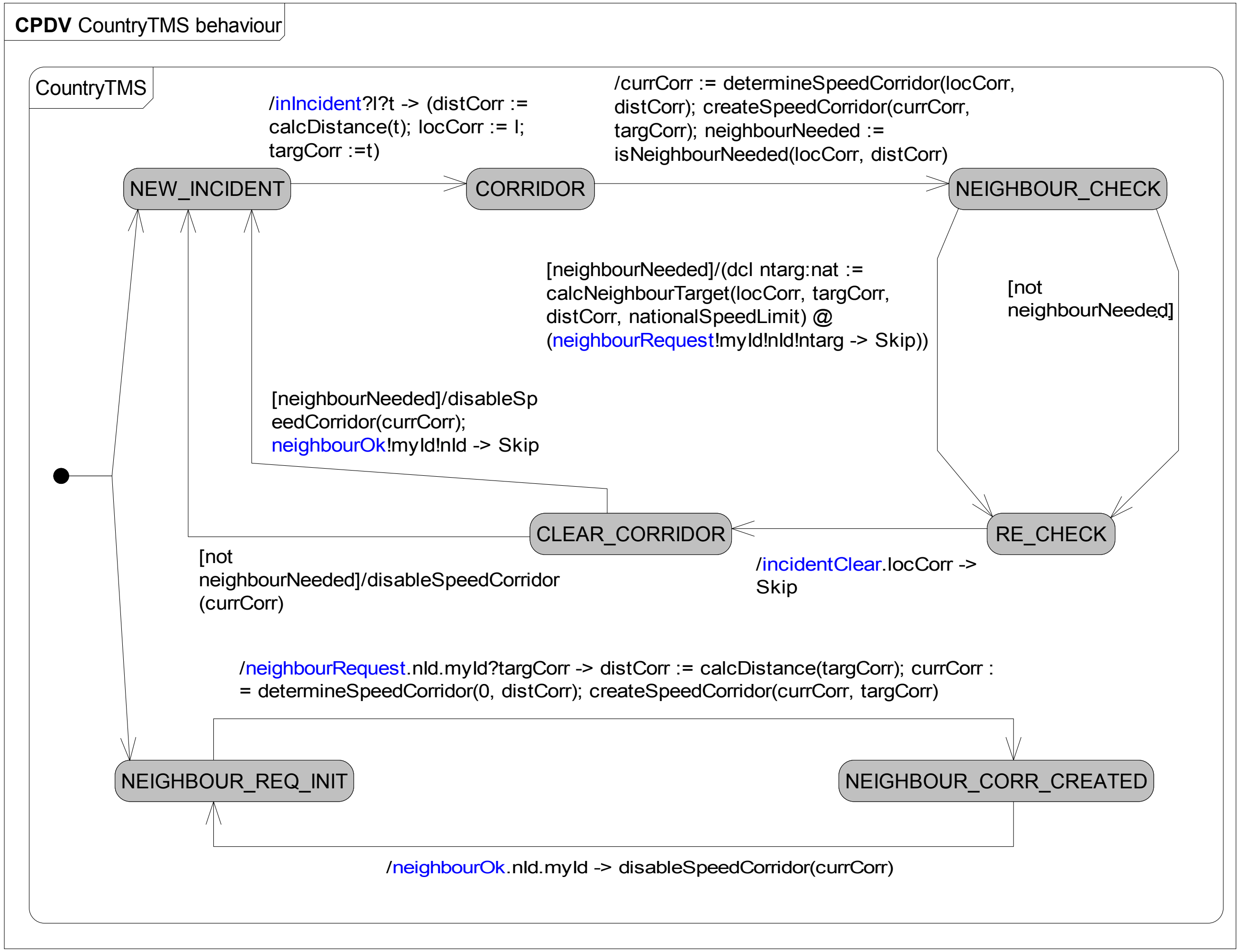
Architectural Modelling

- Defining the functionality of the contract



Architectural Modelling

- Defining behaviour and communications



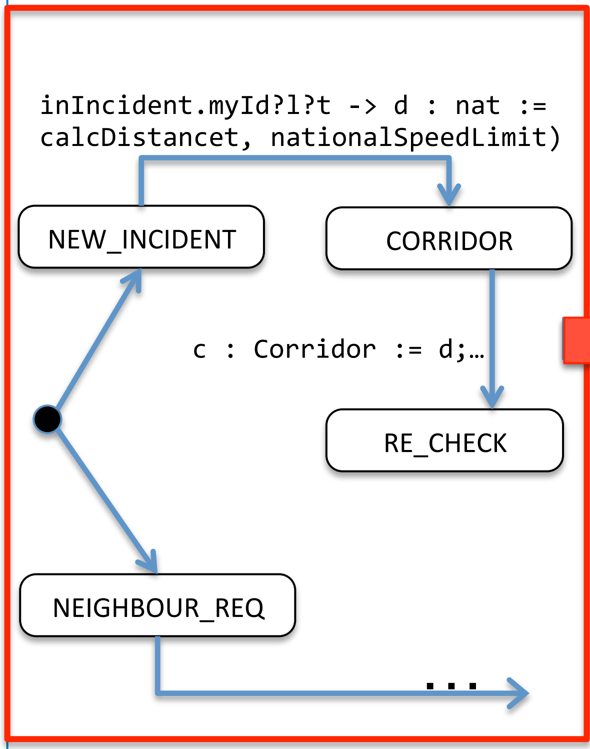
Formal Modelling

- CML – COMPASS Modelling Language – developed for modelling SoS
- Can model data, functionality, event ordering and communication
  - extensible
- Range of formal analysis techniques
- Tools developed for translating models from SysML into CML

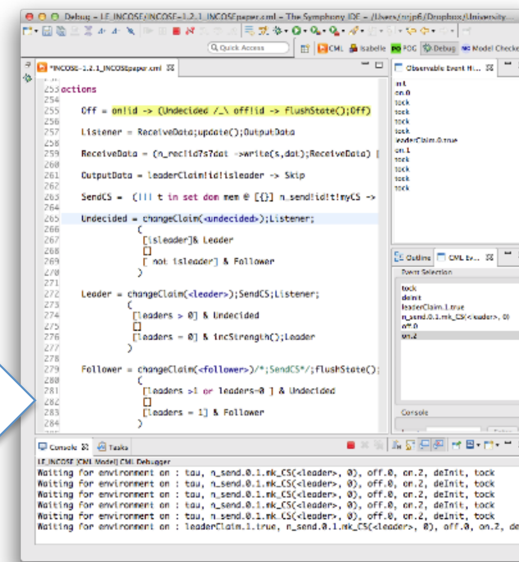
```
process CountryTMS =
begin
  …
  actions
  BEHAVIOUR= NEW_INCIDENT
             []
             NEIGHBOUR_REQ

   NEW_INCIDENT = inIncident.myId?l?t ->
         (dcl d : nat := calcDistance
         (t, nationalSpeedLimit) @
                     CORRIDOR(l, t, d))

   CORRIDOR = l : int, t: nat, d:nat
      @ ACT_STATUS;c:Corridor :=det; …
…
   @ BEHAVIOUR
End

process CountryA = CountryTMS(theAId,
            theBId, limitA, actCorrA)
process CountryB = CountryTMS(theBId,
            theAId, limitB, actCorrB)

process BorderTrafficSoS =
             CountryA [|interface|]
             CountryB
```

Newcastle University

Analysing the Model




```
process CountryTMS =
begin
 …
 actions
 BEHAVIOUR= NEW_INCIDENT
            []
            NEIGHBOUR_REQ

  NEW_INCIDENT = inIncident.myId?l?t ->
        (dcl d : nat := calcDistance
        (t, nationalSpeedLimit) @
                     CORRIDOR(l, t, d))

  CORRIDOR = l : int, t: nat, d:nat
      @ ACT_STATUS;c:Corridor :=det; …
..
  @ BEHAVIOUR
End

process CountryA = CountryTMS(theAId,
            theBId, limitA, actCorrA)
process CountryB = CountryTMS(theBId,
            theAId, limitB, actCorrB)

process BorderTrafficSoS =
             CountryA [|interface|]
             CountryB
```



Symphony Tool Platform

- Analyse cross-border emergent behaviour
- Simulate execution of model
- Proof obligations generated
- Theorem proving

Newcastle University

25th anniversary annual INCOSE international symposium
Seattle, WA
July 13 - 16, 2015

The Value of the Formal Model

Rapid Prototyping

- Replace pre/post contract by an executable version
- Enables simulation of the SoS
- Early validation of requirements

Exploration of design space

- Optimise model choices

Model-based Test

- Derive tests from CPDV and perform against prototype & implementations

```
determineSpeedCorridor (startLoc: int, distance:nat)
resc: Corridor
pre len acts > 1 -- the TMS must have actuators
post elems resc subset inds acts and
      forall a in set elems resc @ (acts(a).loc >= startLoc and
      acts(a).loc <= distance+startLoc)
```

↓

```
determineSpeedCorridor : int * nat ==> Corridor
determineSpeedCorridor (startLoc, distance) ==
( dcl corr : Corridor := [] @
 ( for index = 1 to len acts by 1 do
   ( if (acts(index).loc >= startLoc and
        acts(index).loc <= startLoc+distance)
    then corr := corr ^ [index] );
   return corr ) )
```

The Value of the Formal Model

Contract Verification

- Proof Obligations
  - Operation contracts are satisfiable
  - Invariants are not contradictory
  - Safe applications of operators

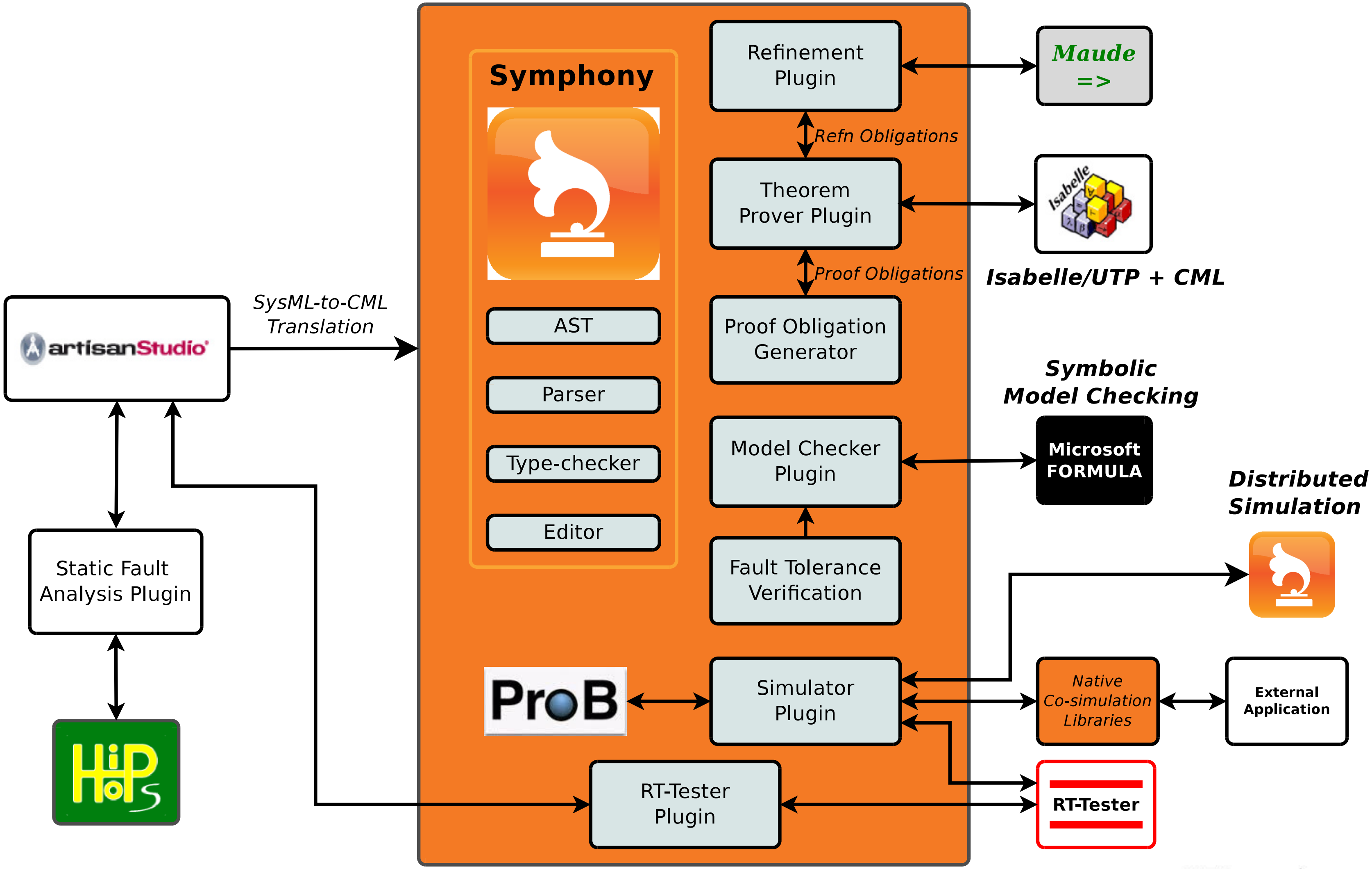
CML PO List

No.	Res.	Type	Source
1	✗	type invariant satisfiable	BorderTrafficv2.cml – GLOBAL
2	✓	non-zero	BorderTrafficv2.cml – GLOBAL
3	✓	type compatibility	BorderTrafficv2.cml – GLOBAL
4	✓	legal function application	BorderTrafficv2.cml – interval
5	✓	non-zero	BorderTrafficv2.cml – GLOBAL
6	✓	non-empty sequence	BorderTrafficv2.cml – GLOBAL
7	✓	legal function application	BorderTrafficv2.cml – GLOBAL
8	✓	non-empty set	BorderTrafficv2.cml – GLOBAL

- Obligations can be discharged with machine support (automated theorem proving).
- But this is not always possible
- Tactics and domain-specific theories increase the level of automation

Symphony Tool Platform

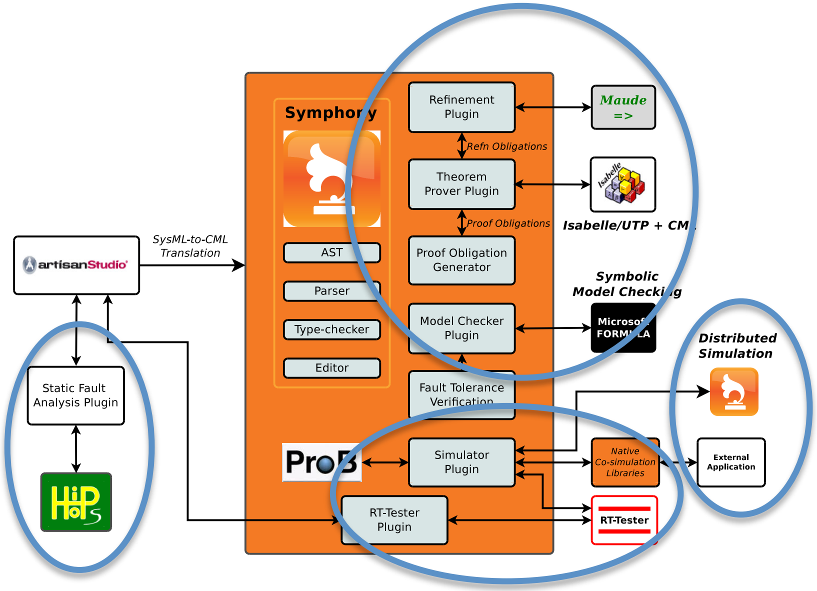
INCOSE 2015



Newcastle University

Symphony Tool Platform

- Interpreter and RT-Tester used for *requirement validation*
- Theorem prover and model checker used for *property verification*
- Static fault analysis allows *FMEA* and *fault tree analysis*
- External links allow distributed SoS engineering



Deployment in SE Processes

Adaptable to general SE process stages, e.g.

- Exploratory Research
  - SoS-ACRE considers stakeholder perspectives at the SoS level, to identify inconsistencies, conflicts or hidden dependencies
- Concept
  - Formal analyses to identify inconsistencies and design problems
  - Supporting trade-space between candidate solutions
  - Models provide design rationale
- Development
  - Supports varied analysis techniques for V&V of requirements, architectural choices, and detailed designs
- Production, Utilisation and Support
  - e.g., assessing proposed changes for unexpected performance degradation or propagated changes

Conclusions

- Proof of concept: we aimed to evaluate the benefits of verification technology at the SoS emergence level.
- SE potential for formal models in analysing emergence
  - Range of analysis techniques enabled (simulation, test, model-check, proof)
  - Quality of evidence and reliance
  - No such thing as a free verification. Cost/benefit trade: hence a stepped approach
- Obtaining a flow from requirements to formalisation
  - Must be supported by the use of a consistent ontology, architectural framework and traceability links
  - We need SE input on realistic problems (scale, emergence challenges, …)

Conclusions

- Tools performance and robustness varies widely
  - Generic solutions less likely to succeed than surgically targeted application
  - Use patterns, frameworks, language subsets
- Largely homogeneous models of the constituent system interfaces
  - Co-modelling allows principled integration of models of digital systems and physics
- More results - on patterns, and more transport studies, Thursday 8 a.m., Grand Ballroom C

Newcastle University

claire.ingram@ncl.ac.uk @_Claire_Ingram

john.fitzgerald@ncl.ac.uk @NclFitz

This work is part of the COMPASS project: research into model-based techniques for developing, maintaining and analysing SoSs

COMPASS

thecompassclub.org