BANG & OLUFSEN

Newcastle University

Maintaining Emergence in Systems of Systems Integration: a Contractual Approach using SysML

Jeremy Bryans; John Fitzgerald; Richard Payne
(Newcastle University, UK)

Klaus Kristensen
(Bang & Olufsen, DK)

July 2015

Overview

- Systems of Systems
- Case study
- Modelling
- Analysis
- Conclusions and future work

Bryans J, Fitzgerald J, Payne R, Kristensen K., 2014. Maintaining Emergence in Systems of Systems Integration: a Contractual Approach using SysML. In: INCOSE International Symposium (IS2014). 2014, Las Vegas, Nevada.

Overview

- **Systems of Systems**
- Case study
- Modelling
- Analysis
- Conclusions and future work

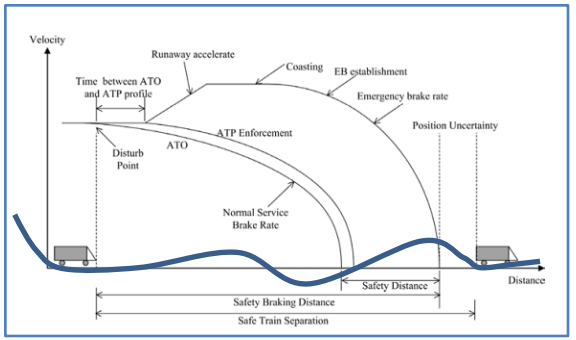
Our Research …

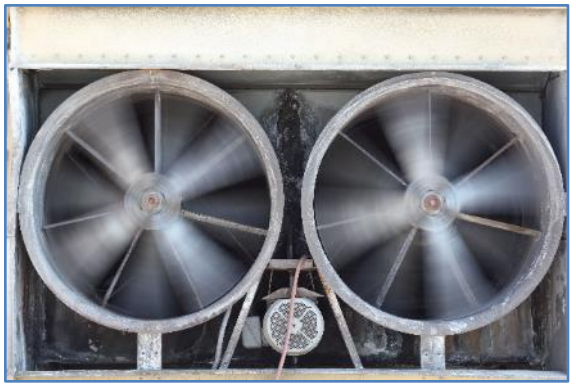
Design technology (foundations, methods, tools) for:

- Systems of Systems (SoS)
- Cyber-Physical Systems (CPS)

We focus on model-based design:

- Models as a basis for collaborative development
- Machine-assisted analysis of models as a means of managing development risk





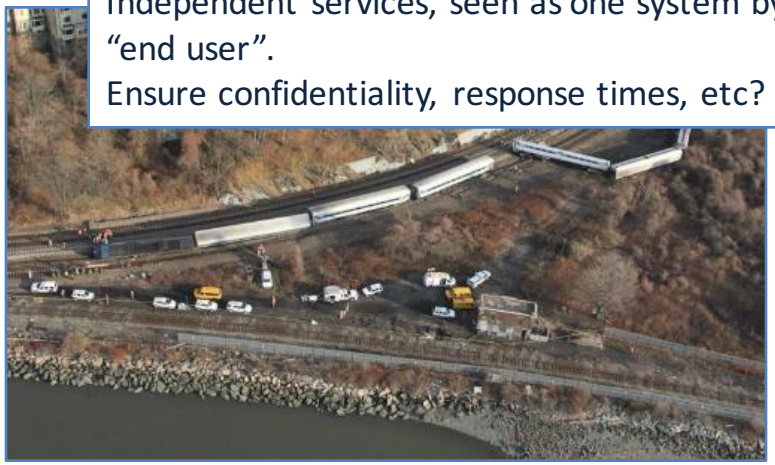
"HVAC Ventilation Exhaust" by PictorialEvidence - Own work. Licensed under CC BY-SA 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:HVAC_Ventilation_Exhaust.jpg#/media/File:HVAC_Ventilation_Exhaust.jpg

Systems of Systems

- **Operational & Managerial Independence of Constituent Systems**
  - Constituent systems evolve independently
- **Complexity of confirming/refuting SoS-level properties**
  - Verification of emergence
- **Semantic heterogeneity (integrating models)**
  - Wide range of interacting features in models (e.g. location, time, concurrency, data, communication)

Audio/Video (Bang & Olufsen)
Independent networks, devices, content services. Ensure a consistent "SoS experience"

Emergency Response (Insiel)
Independent services, seen as one system by "end user".
Ensure confidentiality, response times, etc?

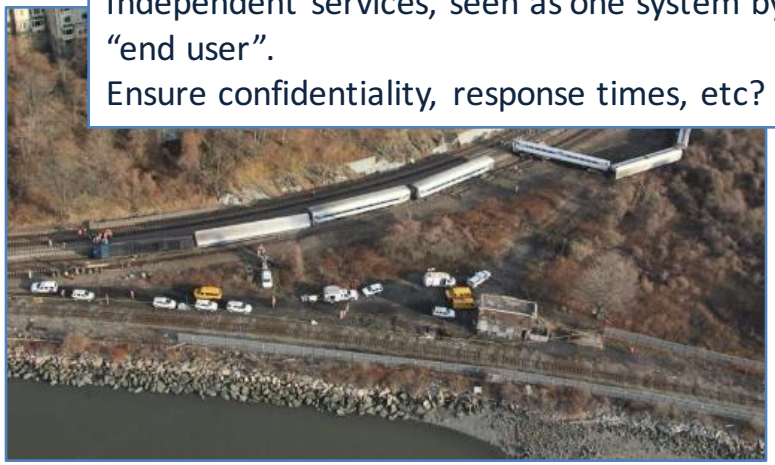


Systems of Systems

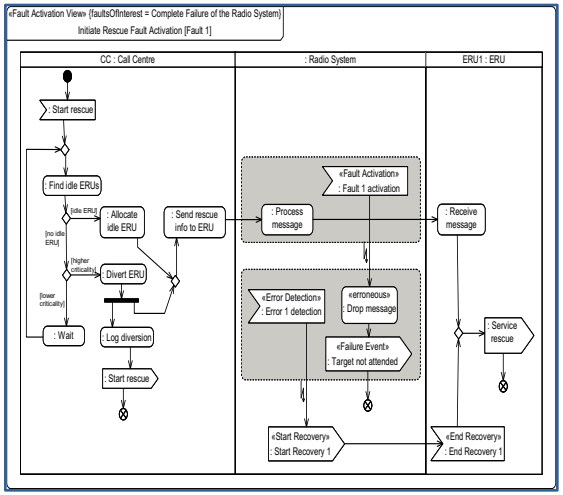
- **Operational & Managerial Independence of Constituent Systems**
  - Constituent systems evolve independently
- **Complexity of confirming/refuting SoS-level properties**
  - Verification of emergence
- **Semantic heterogeneity (integrating models)**
  - Wide range of interacting features in models (e.g. location, time, concurrency, data, communication)

Audio/Video (Bang & Olufsen)
Independent networks, devices, content services. Ensure a consistent "SoS experience"

Emergency Response (Insiel)
Independent services, seen as one system by "end user".
Ensure confidentiality, response times, etc?



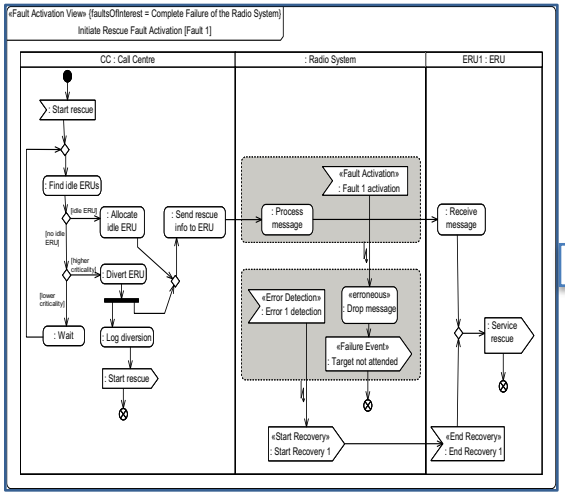
COMPASS Technology



SysML modelling

- Guidelines for Requirements, Architecture, Integration
- SoS Modelling profiles, e.g. Fault-Error-Failure
- Architectural patterns and extensible frameworks

COMPASS Technology




```
actions
 MERGE1(r) =
  (dcl e: set of ERUId @ e := findIdleERUs();
   (do
     e = {} -> DECISION2(r)
    |
     e <> {} ->
      (dcl e1: ERUId @ e1 :=
        allocateIdleERU(e, r); MERGE2(e1, r))
   end)) …
process InitiateRescue = CallCentreProc
  [| SEND_CHANNELS |] RadioSystemProc
  [| RCV_CHANNELS |] ERUsProc
```

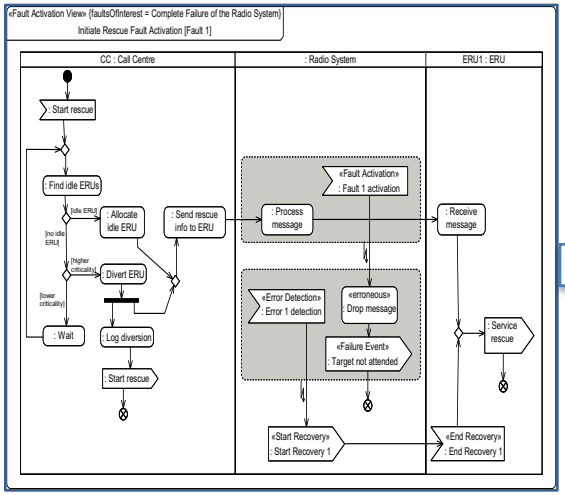
SysML modelling

- Guidelines for Requirements, Architecture, Integration
- SoS Modelling profiles, e.g. Fault-Error-Failure
- Architectural patterns and extensible frameworks

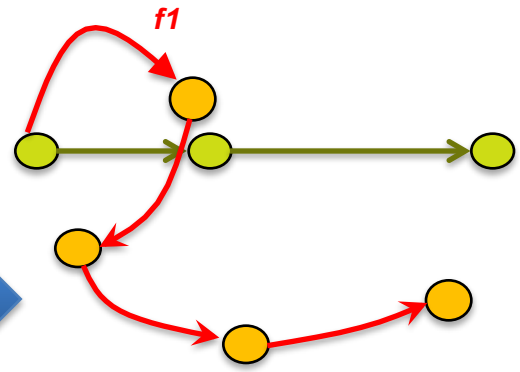
Formal Modelling Language

- CML allows representation of behavioural semantics of the SoS
- Supports contract specification
- Describes functionality, object-orientation, concurrency, real-time, mobility.
- Can be extended to new paradigms

COMPASS Technology




```
actions
 MERGE1(r) =
  (dcl e: set of ERUId @ e := findIdleERUs();
   (do
     e = {} -> DECISION2(r)
     |
     e <> {} ->
      (dcl e1: ERUId @ e1 :=
        allocateIdleERU(e, r); MERGE2(e1, r))
   end)) …
process InitiateRescue = CallCentreProc
  [| SEND_CHANNELS |] RadioSystemProc
  [| RCV_CHANNELS |] ERUsProc
```



(SoS || STOP) [= $\mathcal{L}_E$(SoS)

Symphony

SysML modelling

- Guidelines for Requirements, Architecture, Integration
- SoS Modelling profiles, e.g. Fault-Error-Failure
- Architectural patterns and extensible frameworks

Formal Modelling Language

- CML allows representation of behavioural semantics of the SoS
- Supports contract specification
- Describes functionality, object-orientation, concurrency, real-time, mobility.
- Can be extended to new paradigms

Tool-supported Analysis

- Model-checker
- Automated proof
- Test generation
- Simulation
- Model-in-Loop Test
- Exploration of design space

Overview

- Systems of Systems
- **Case study**
- Modelling
- Analysis
- Conclusions and future work

AV SoS Case Study

- CSs are *heterogeneous* and may *evolve* (through software or firmware upgrades)
- New CSs may be *integrated* into SoS at any time
- CSs may be *legacy* or *non-B&O systems*

AV SoS Case Study

- Challenge: verifying emergence – can a single "leader" be established to maintain global clock, SoS architecture, streaming details, …?

Overview

- Systems of Systems
- Case study
- **Modelling**
- Analysis
- Conclusions and future work

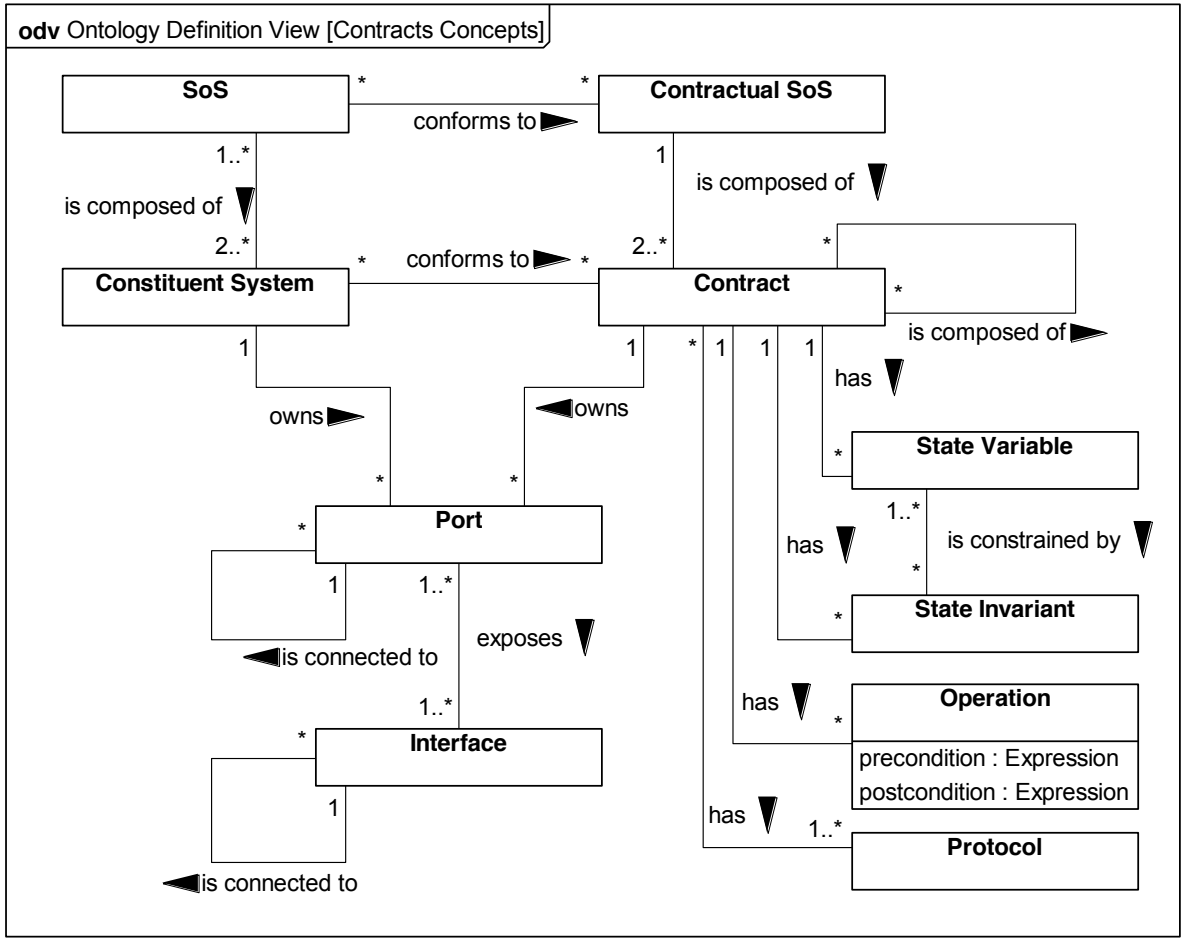
Why Contracts May Help

- SoSs present significant challenges
  - Bound behaviours that can be relied upon *without over-constraining them*
  - *Promote desirable* and *limit undesirable* emergent behaviours
- *Contractual description* of CSs
  - Contract is a description of the "minimum" behaviour that a CS must exhibit in order to be part of an SoS
  - CSs free to choose the way in which they meet these contracts
  - Free to adhere to other contracts
- We present a definition of a contract as a SysML pattern

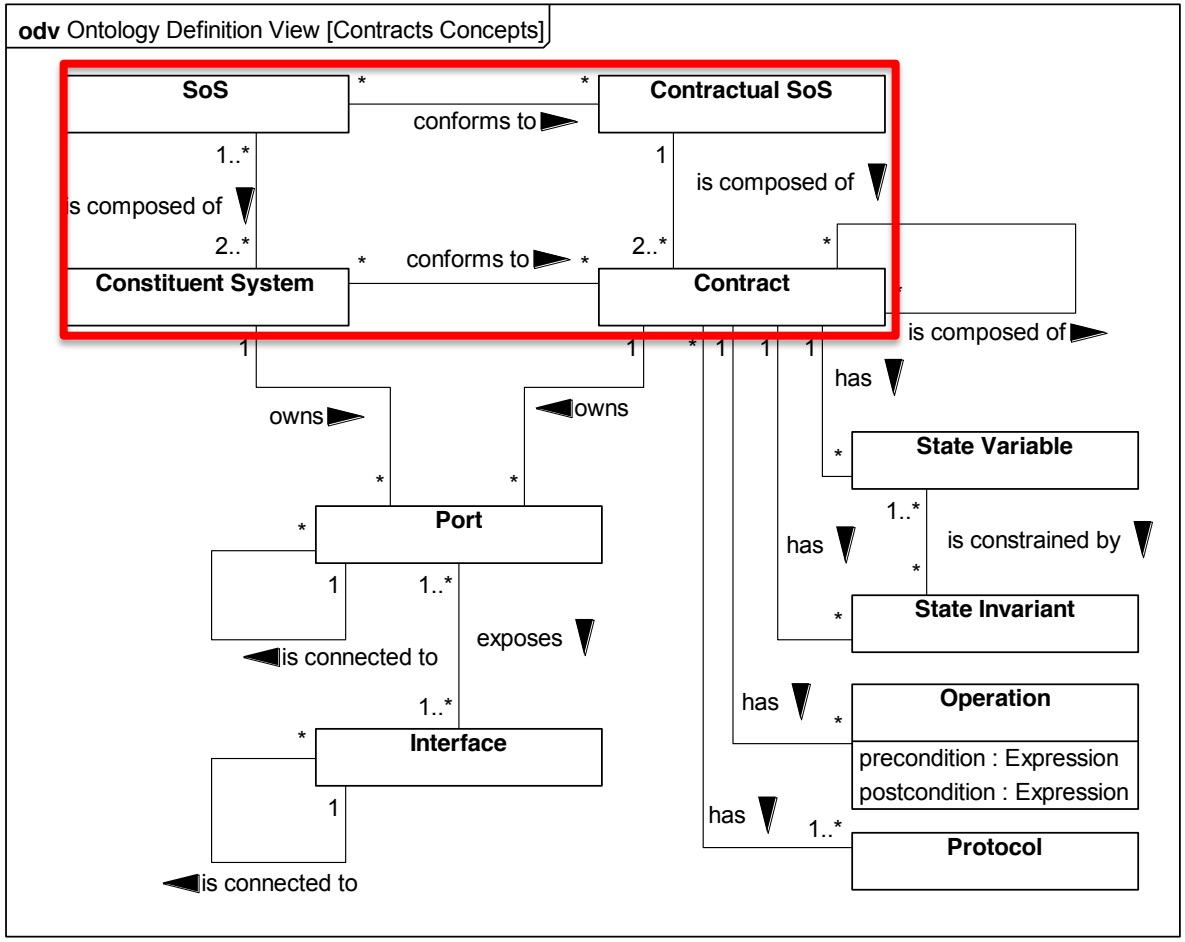
The Contract Pattern

- Collection of viewpoints for modelling and defining the contracts of a SoS
- Defined using SysML and implemented as a SysML profile
- Use a modelling framework for defining patterns:
  - Pattern is a set of related *viewpoints;* each with an identified *context*, *syntax* (permitted modelling elements), and constraining *rules*.
- Notation agnostic
  - Views defined in any notation

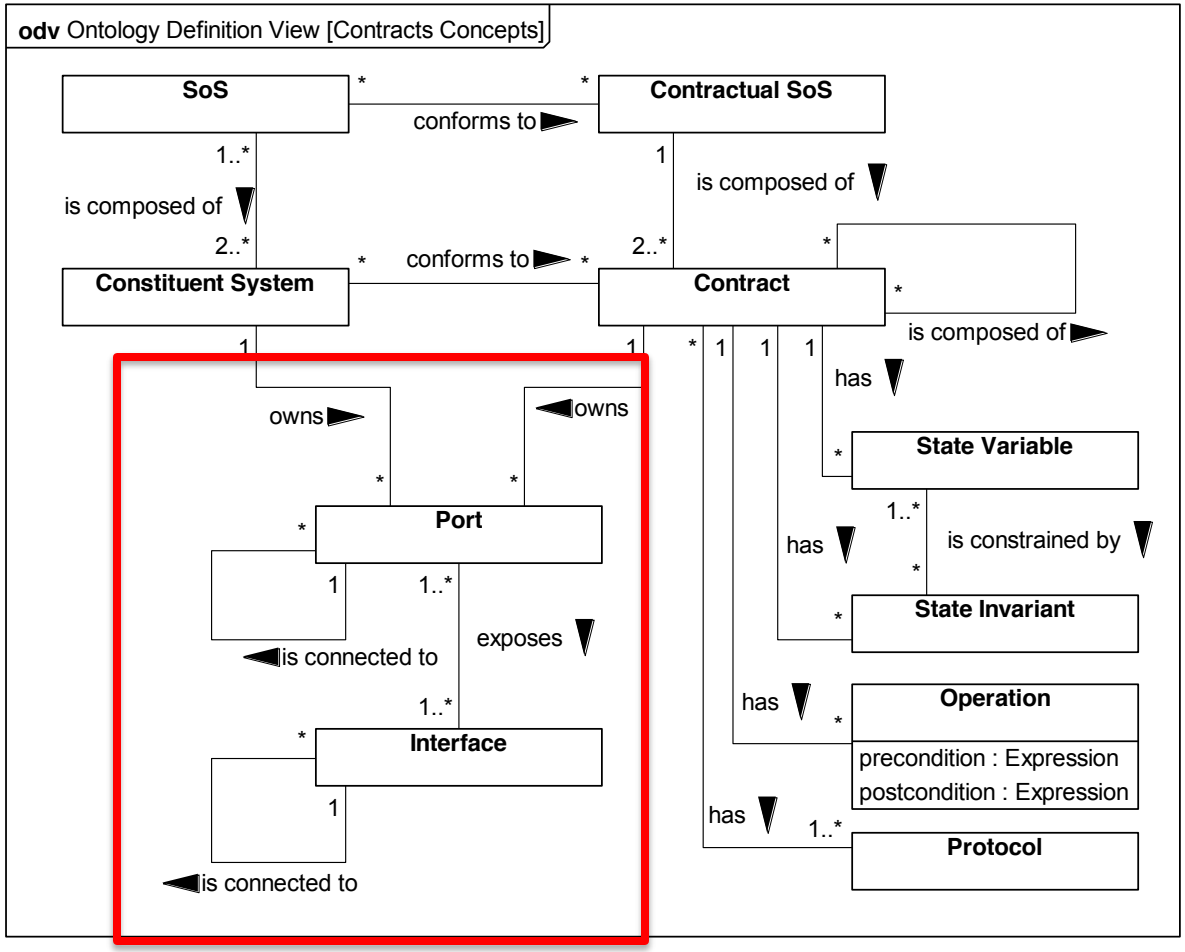
Contract Pattern - Ontology



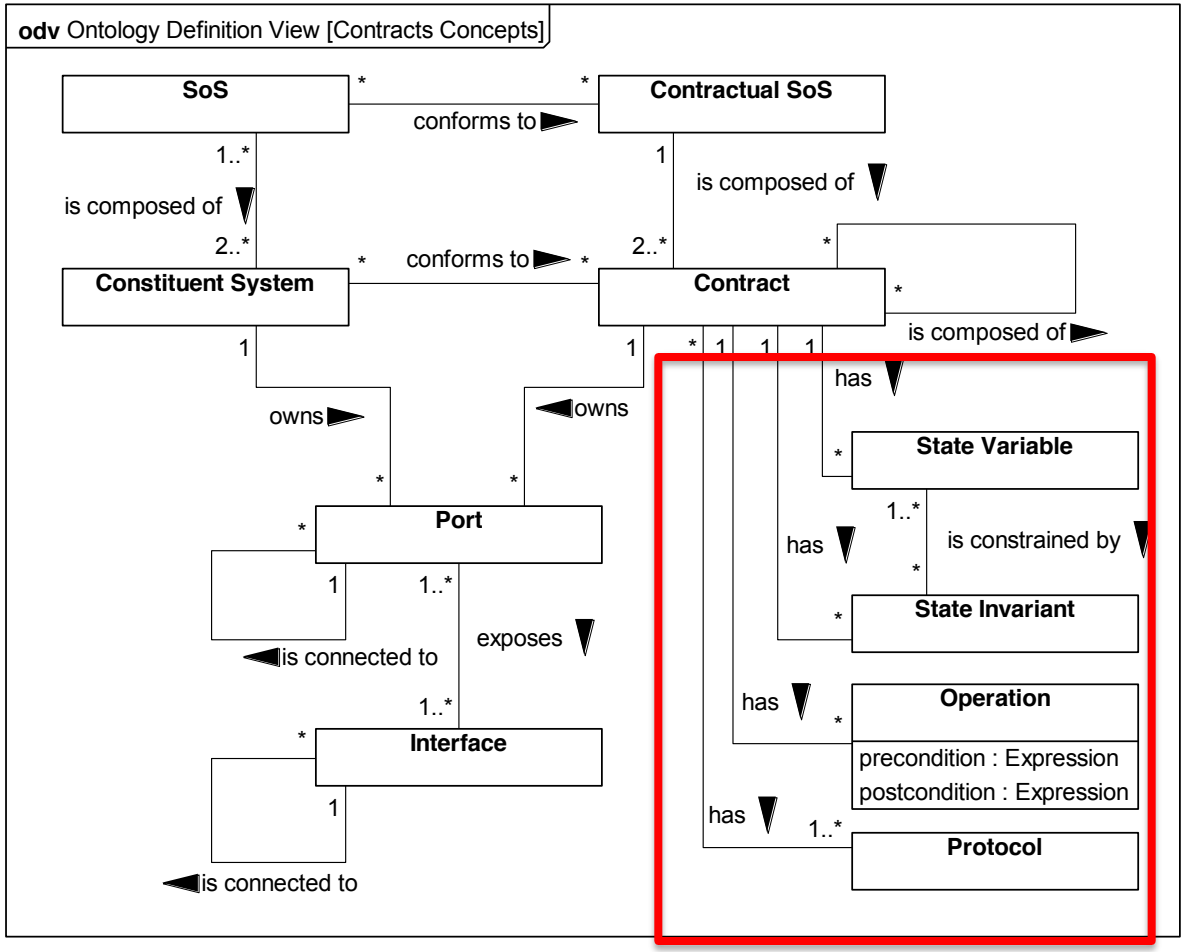
Contract Pattern - Ontology



Contract Pattern - Ontology



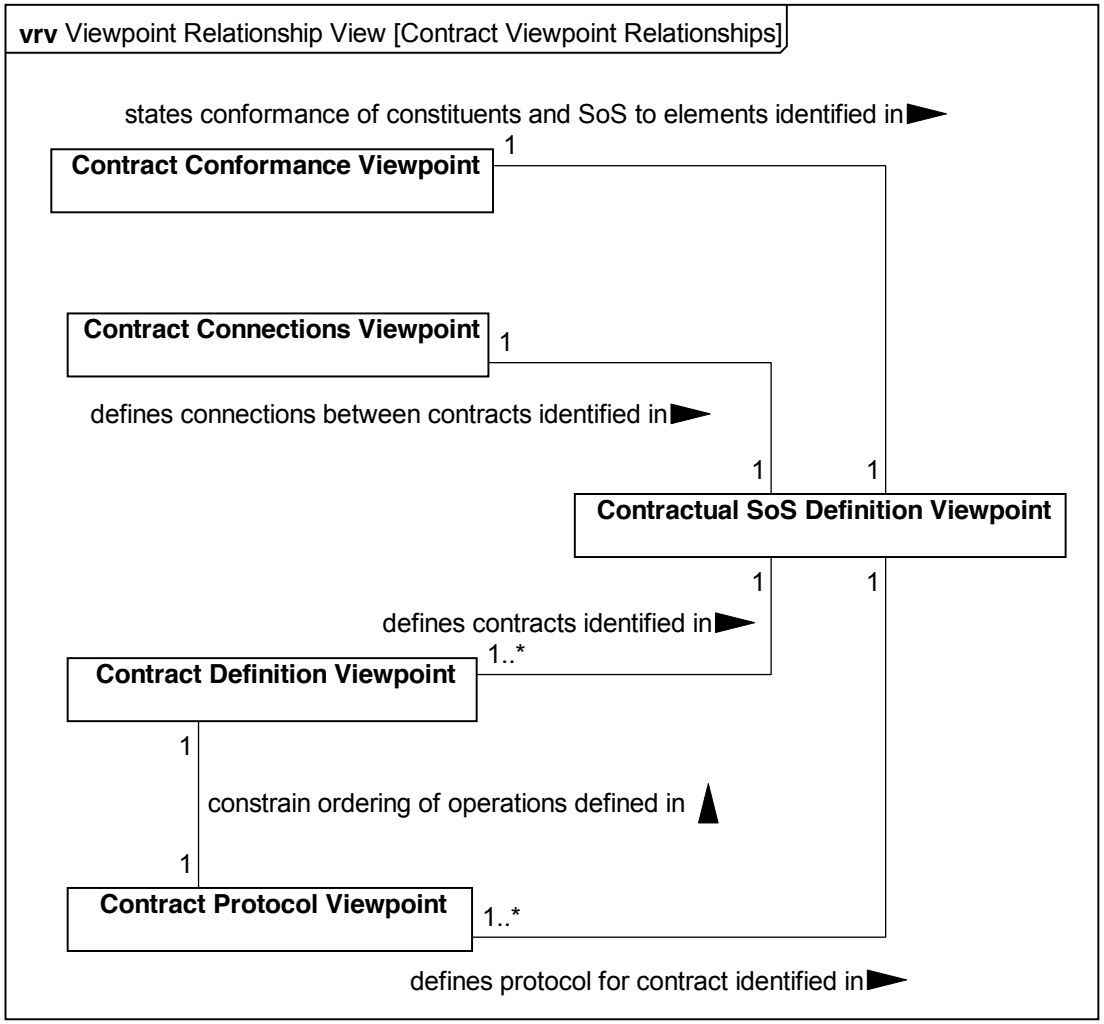
Contract Pattern - Ontology



Contract Pattern - Views

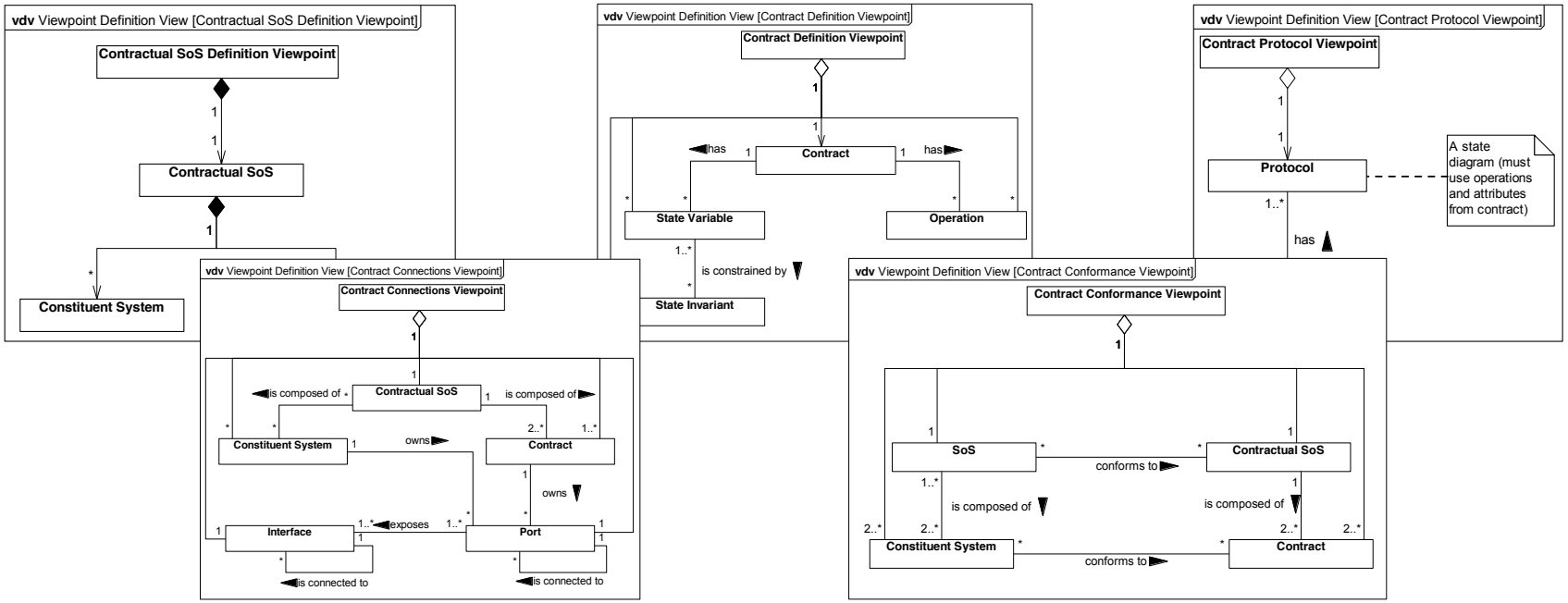
Contract Pattern Viewpoints	
Name	*Overview*
Contractual SoS Definition	Identifies the contracts which comprise the Contractual SoS
Contract Conformance	Denotes the contracts each CS conforms to
Contract Connections	Shows connections and interfaces between contracts
Contract Definition	Defines operations, state variables and state invariants of a contract
Contract Protocol	Protocol specification of a contract

Contract Pattern – Viewpoint Relationships



Contract Pattern – Viewpoint Definition

- Define the model elements on a view and their relationships
- Consistent with ontology

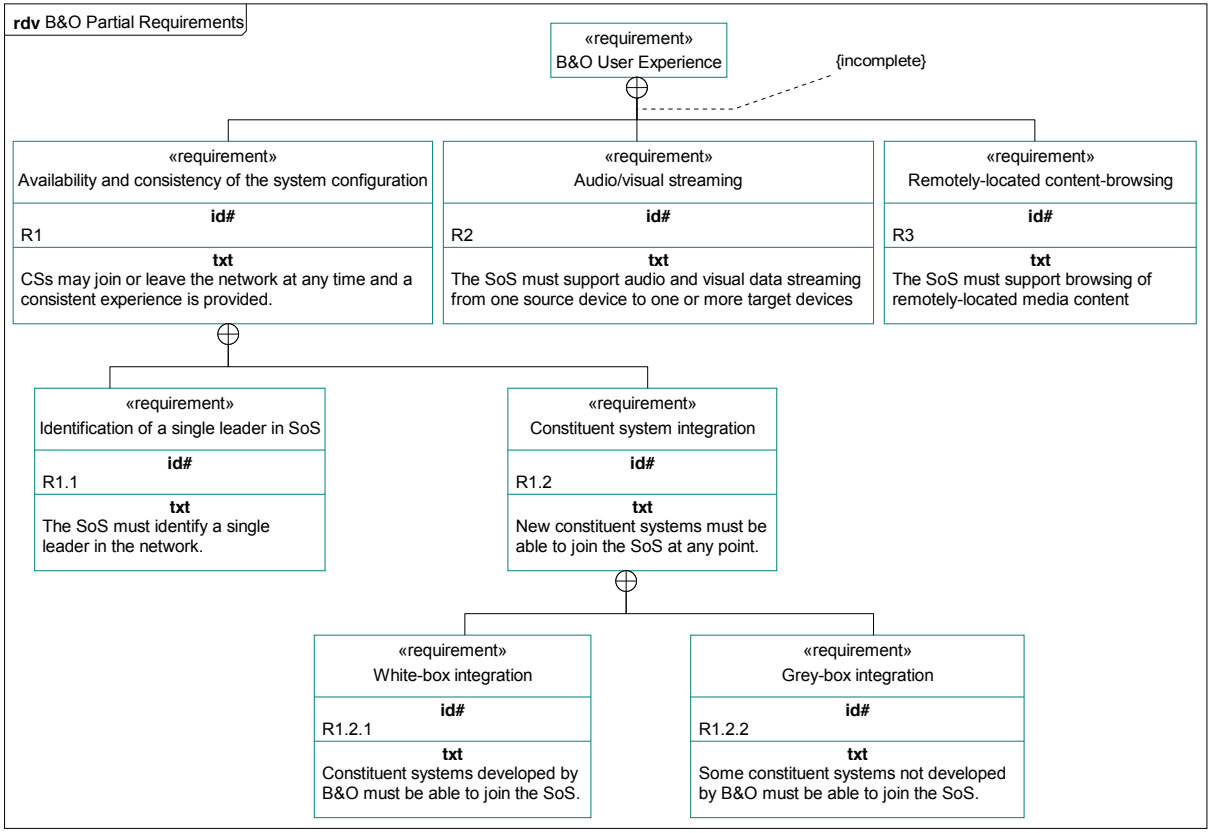


Case Study Revisited

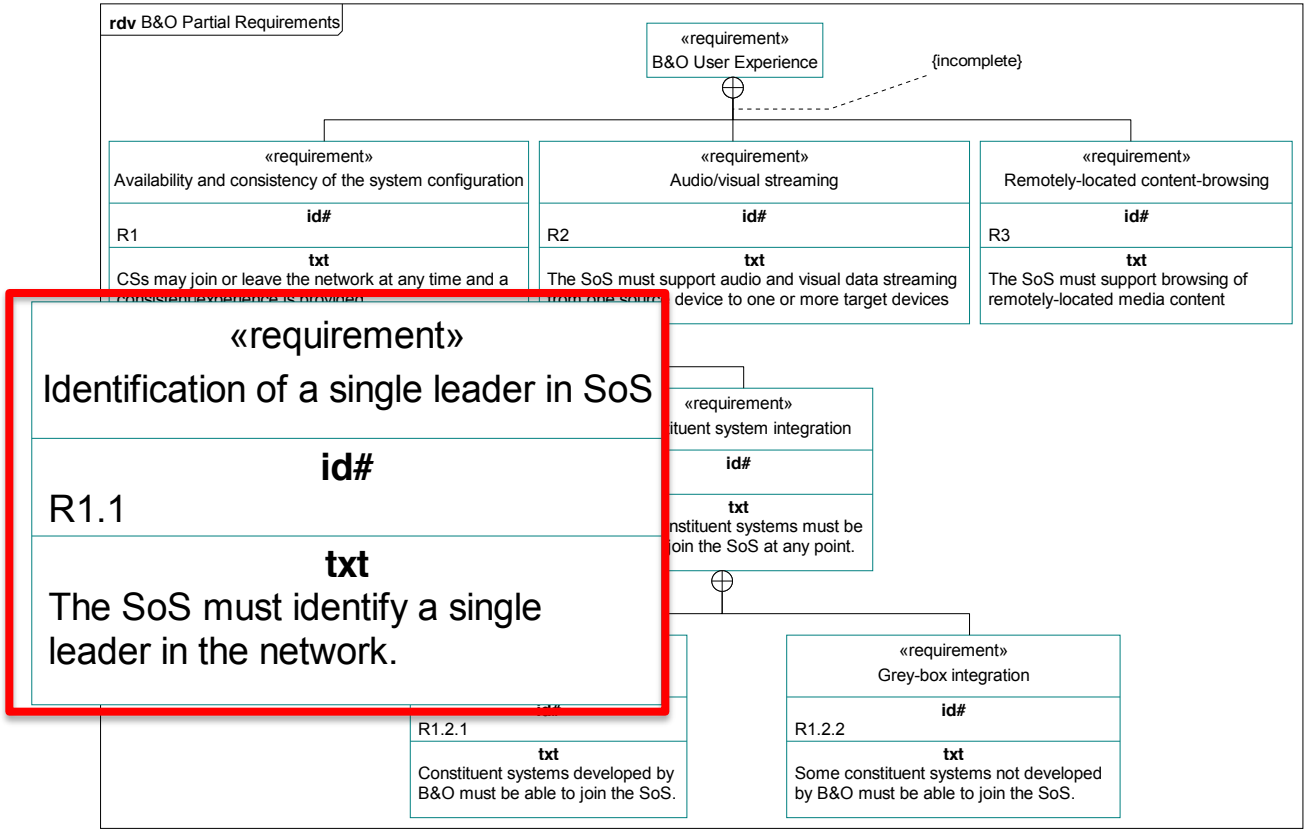
- Based on a Bang & Olufsen (B&O) home Audio Visual (AV) network linking multiple AV devices.

- Challenge: verifying emergence – can a single "leader" be established to maintain global clock, SoS architecture, streaming details, …?

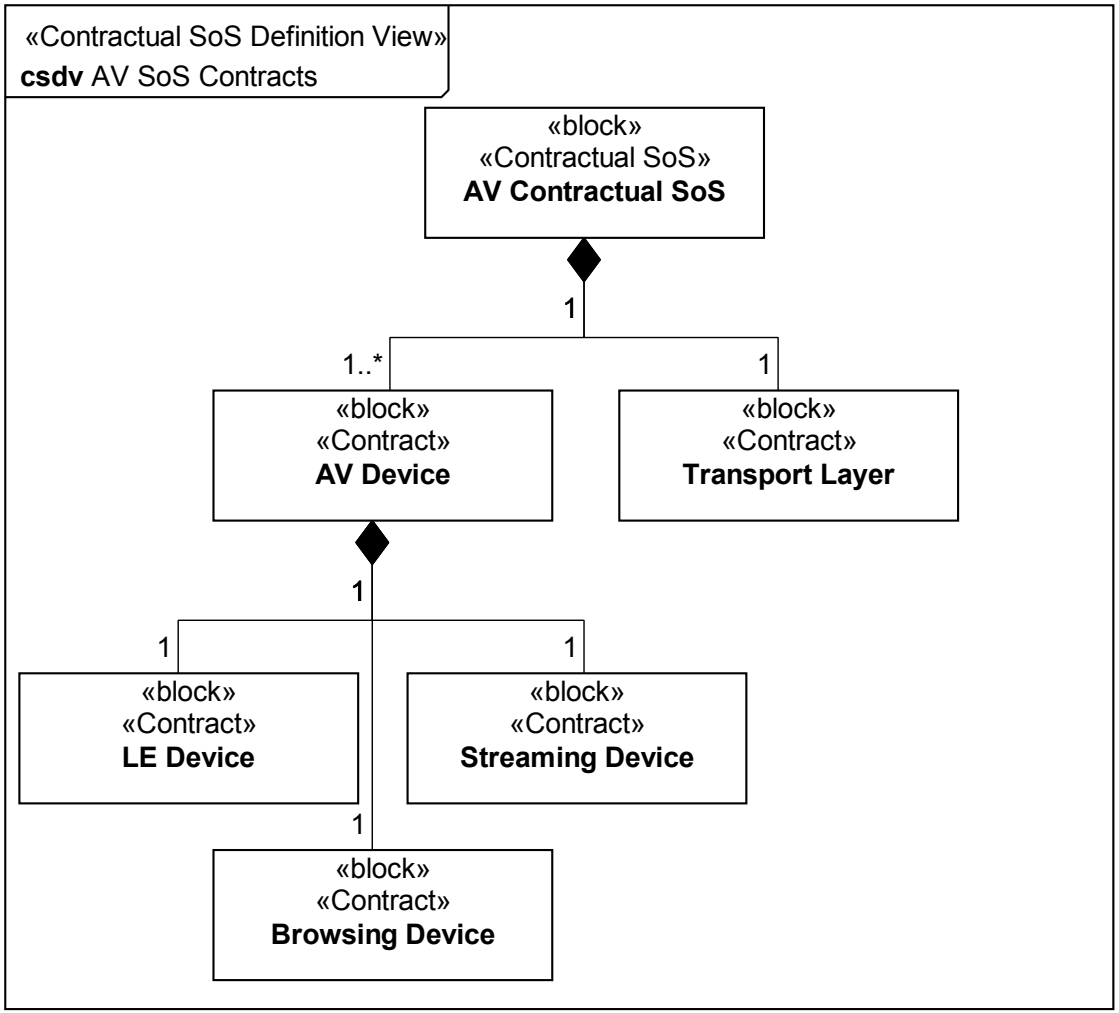
Requirements Definition



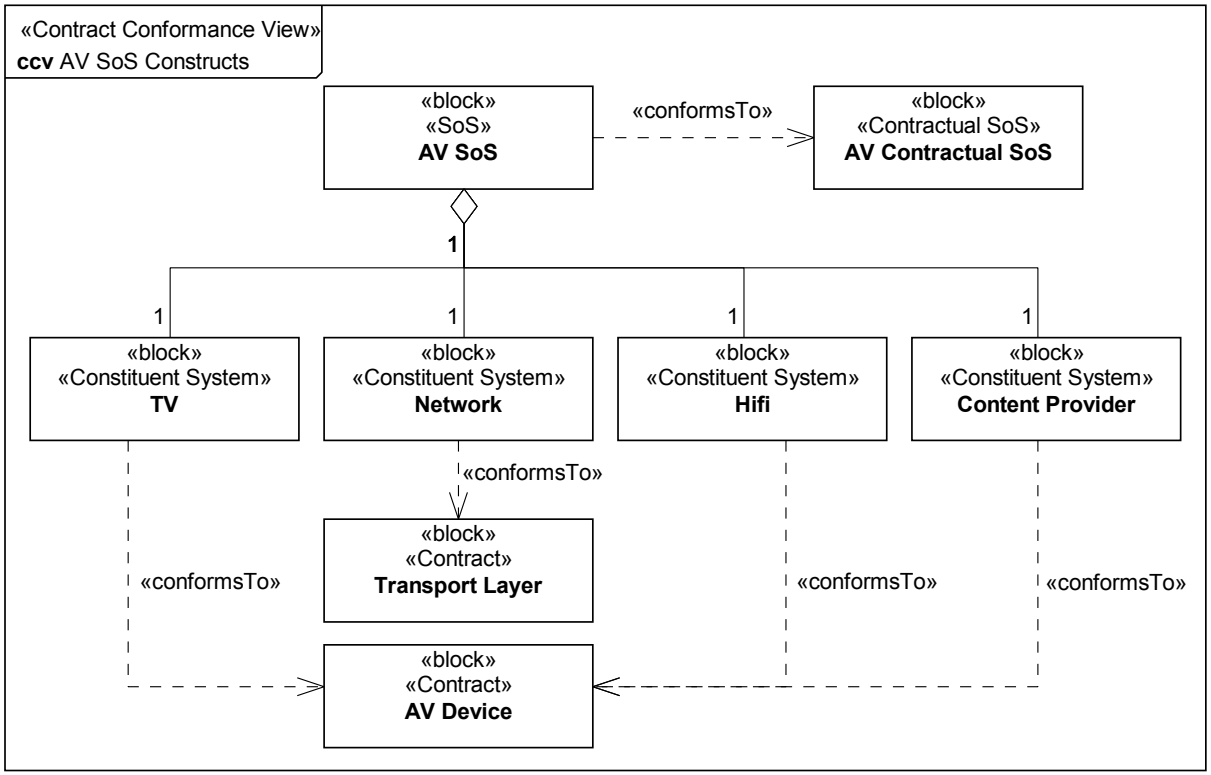
Requirements Definition



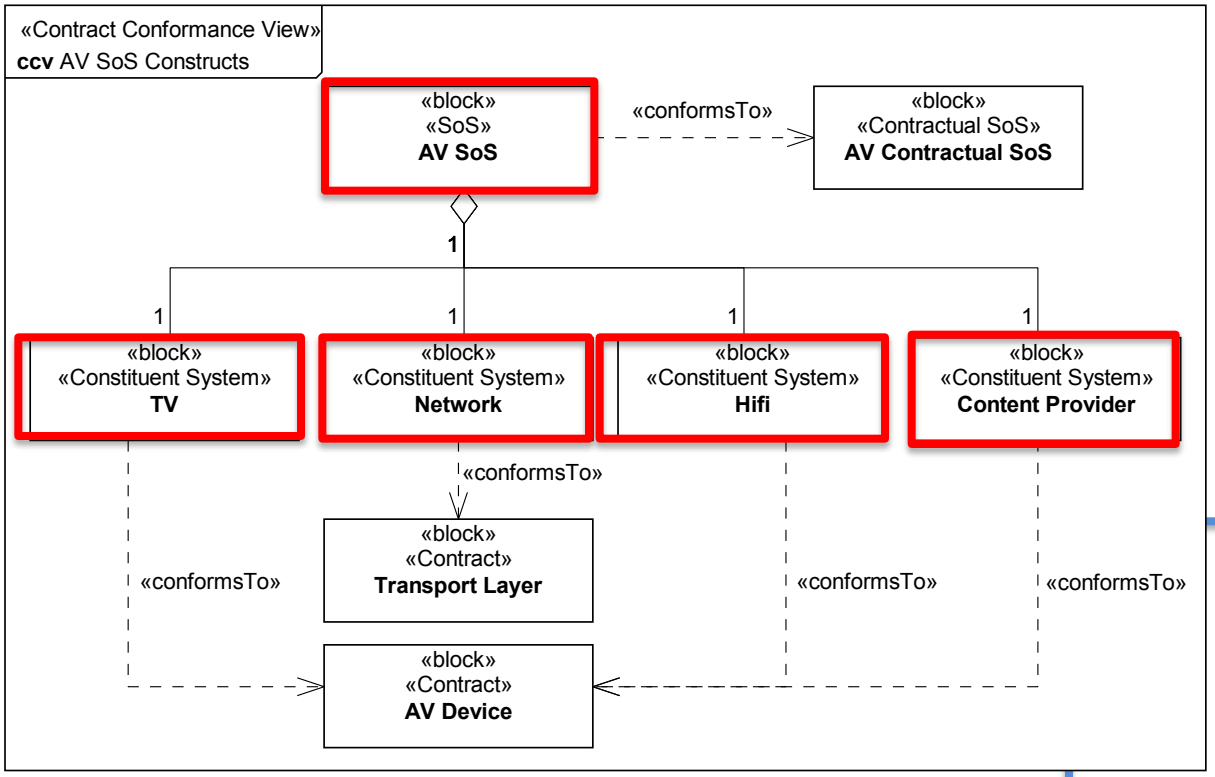
Contractual SoS Definition View

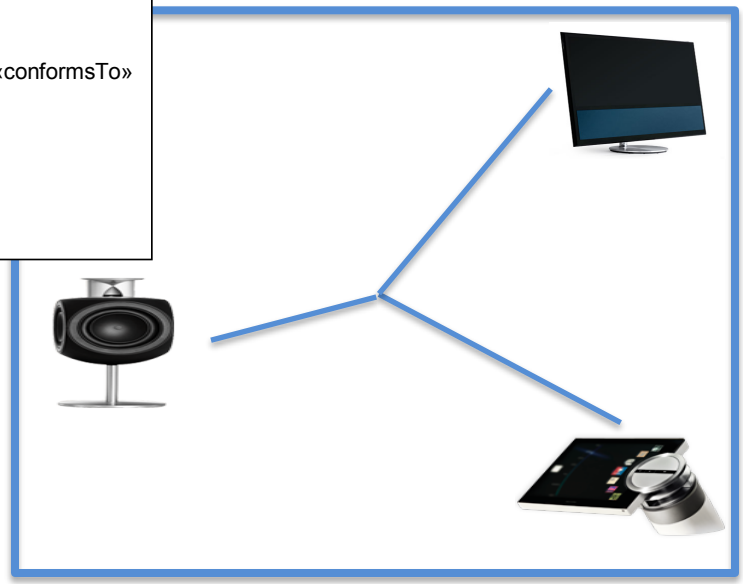


Contract Conformance View

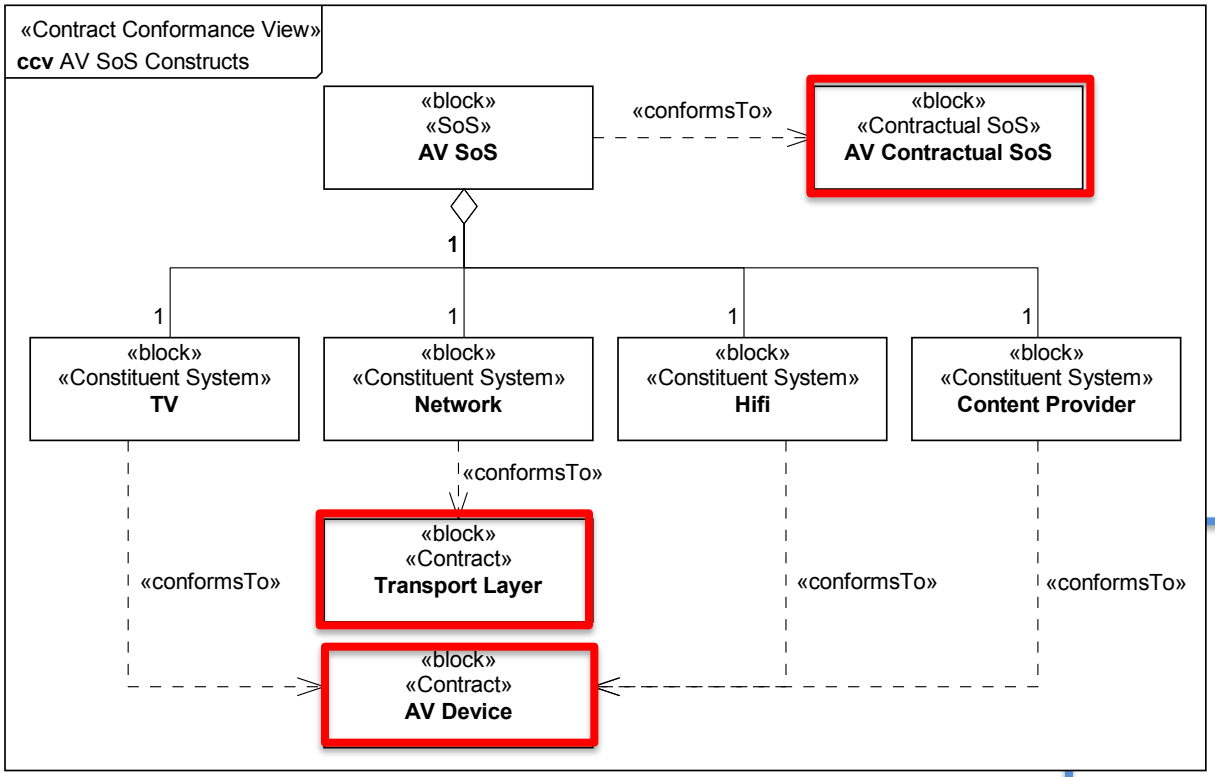


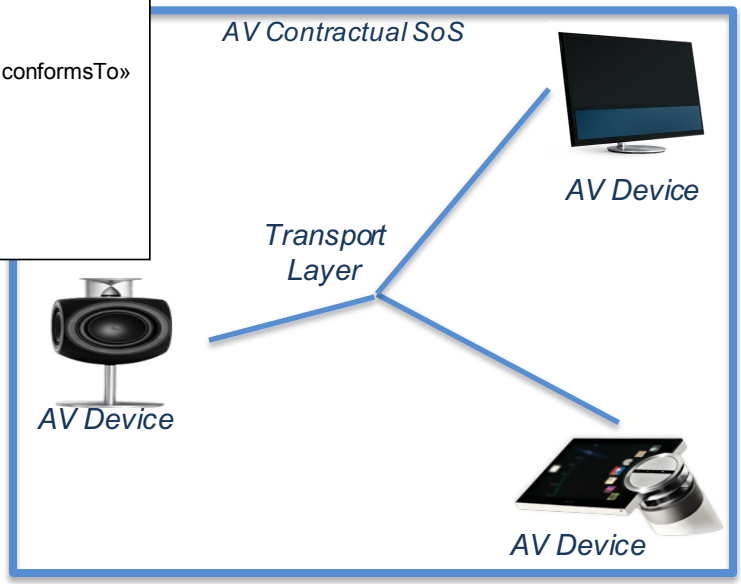
Contract Conformance View



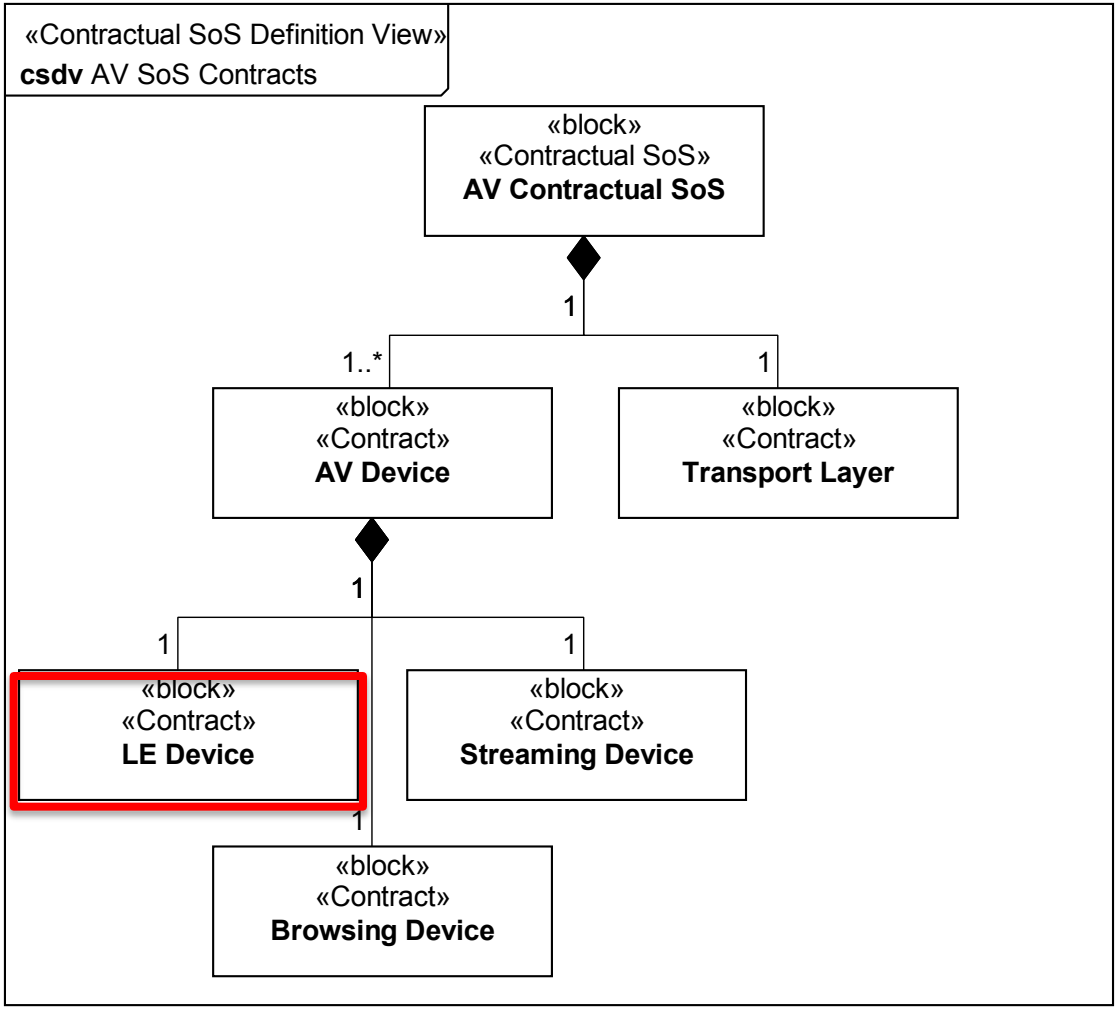


Contract Conformance View



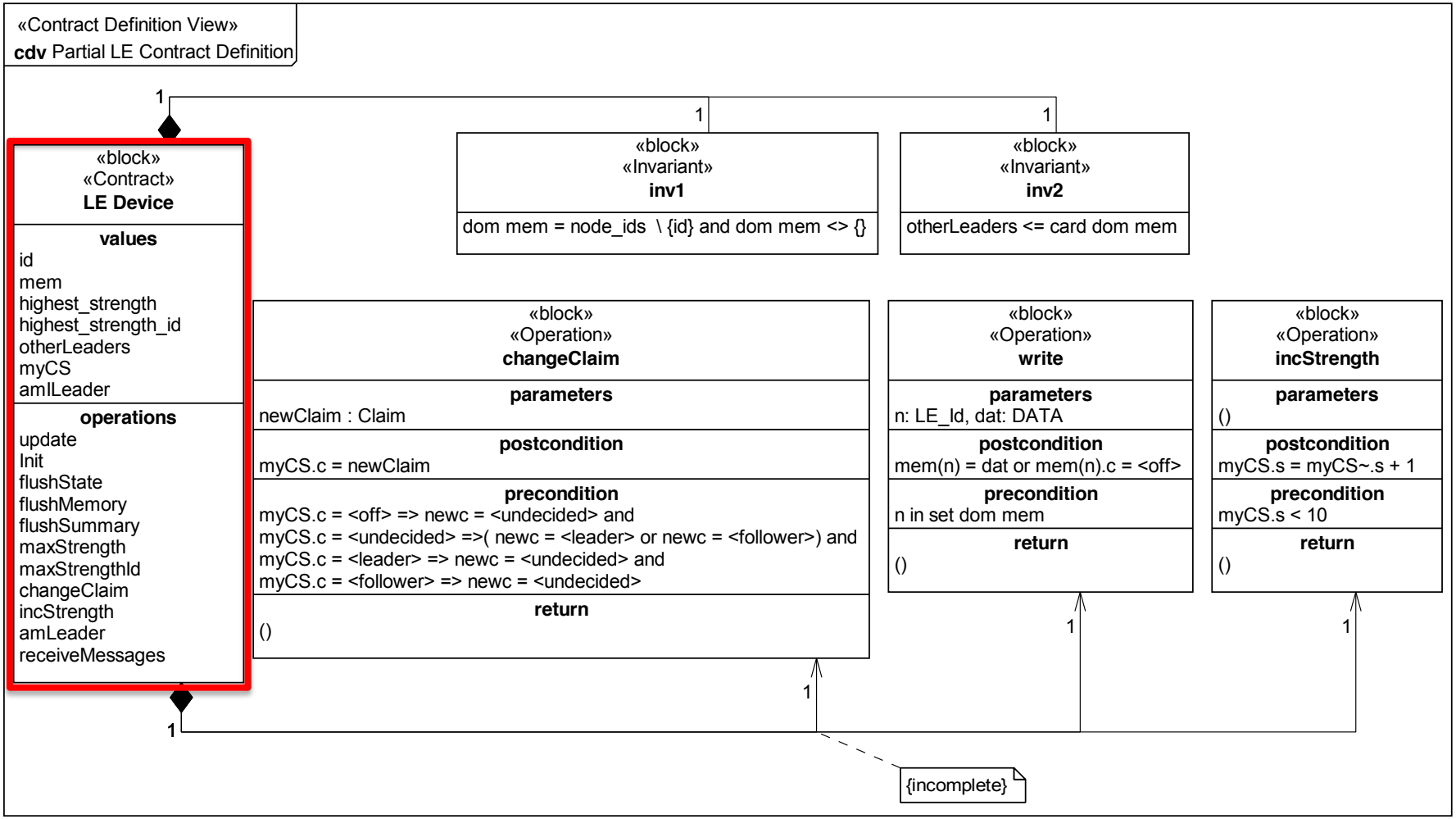


Defining a Contract

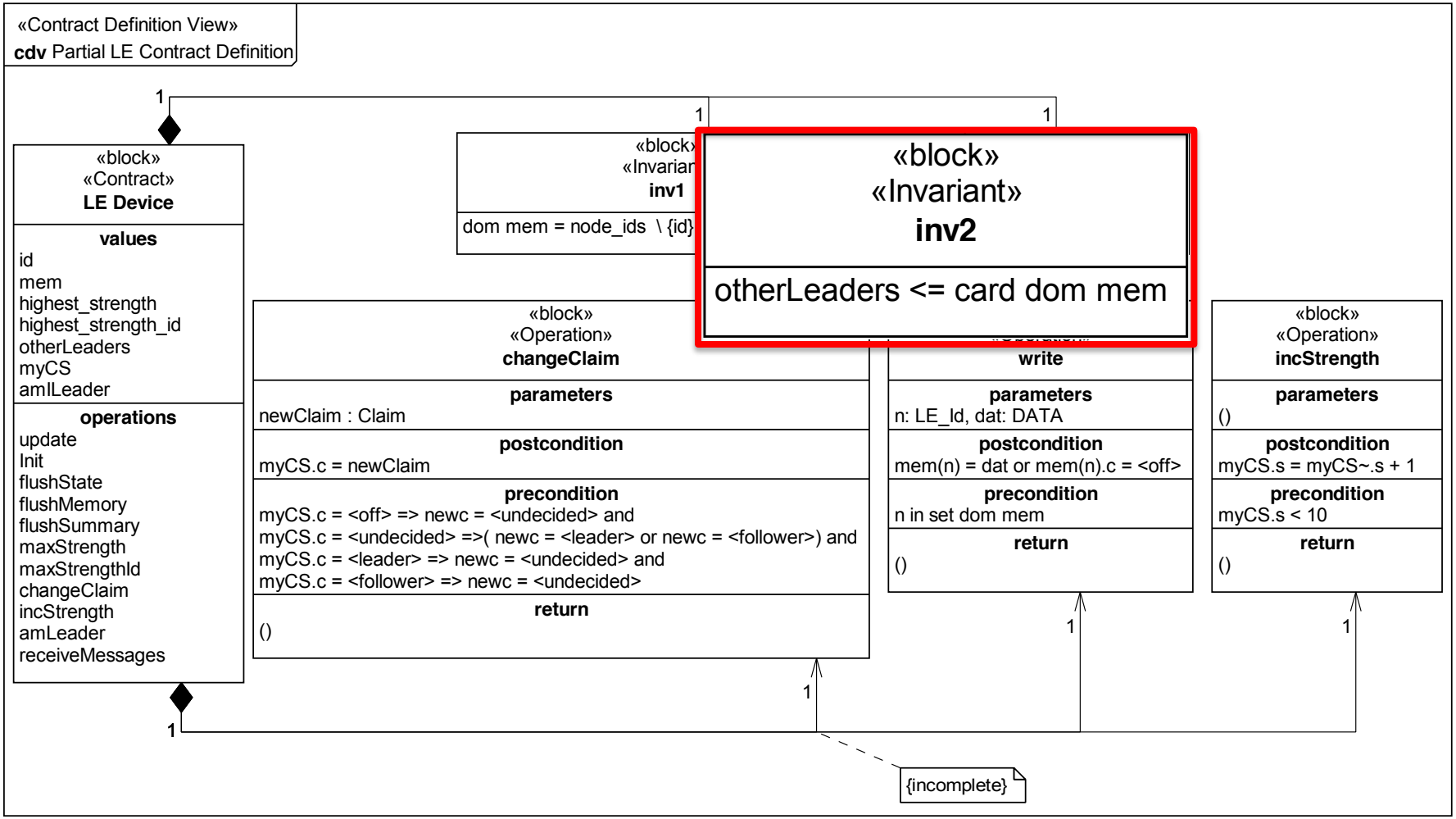


LE Device Contract Definition View

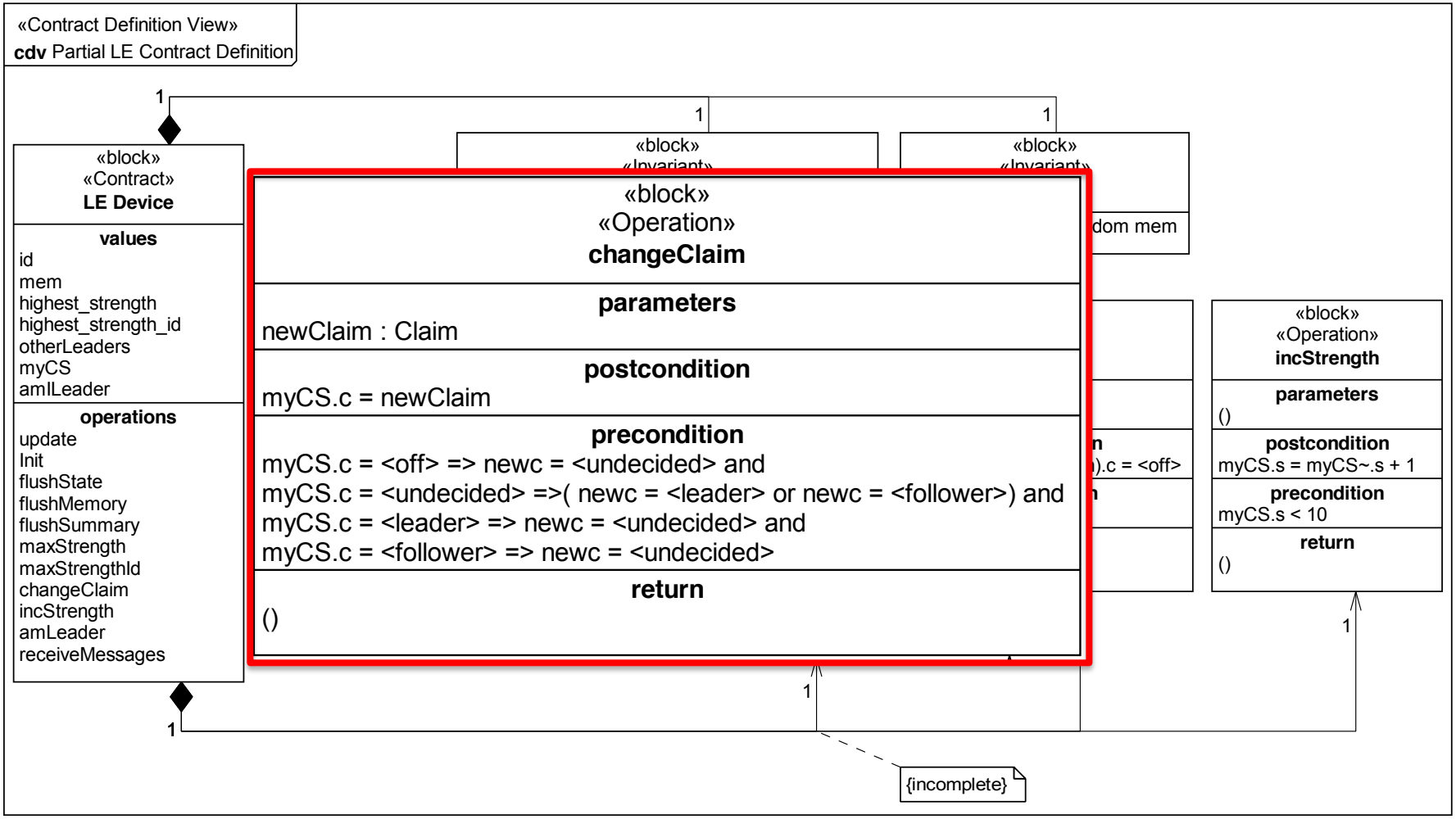
LE Device Contract Definition View



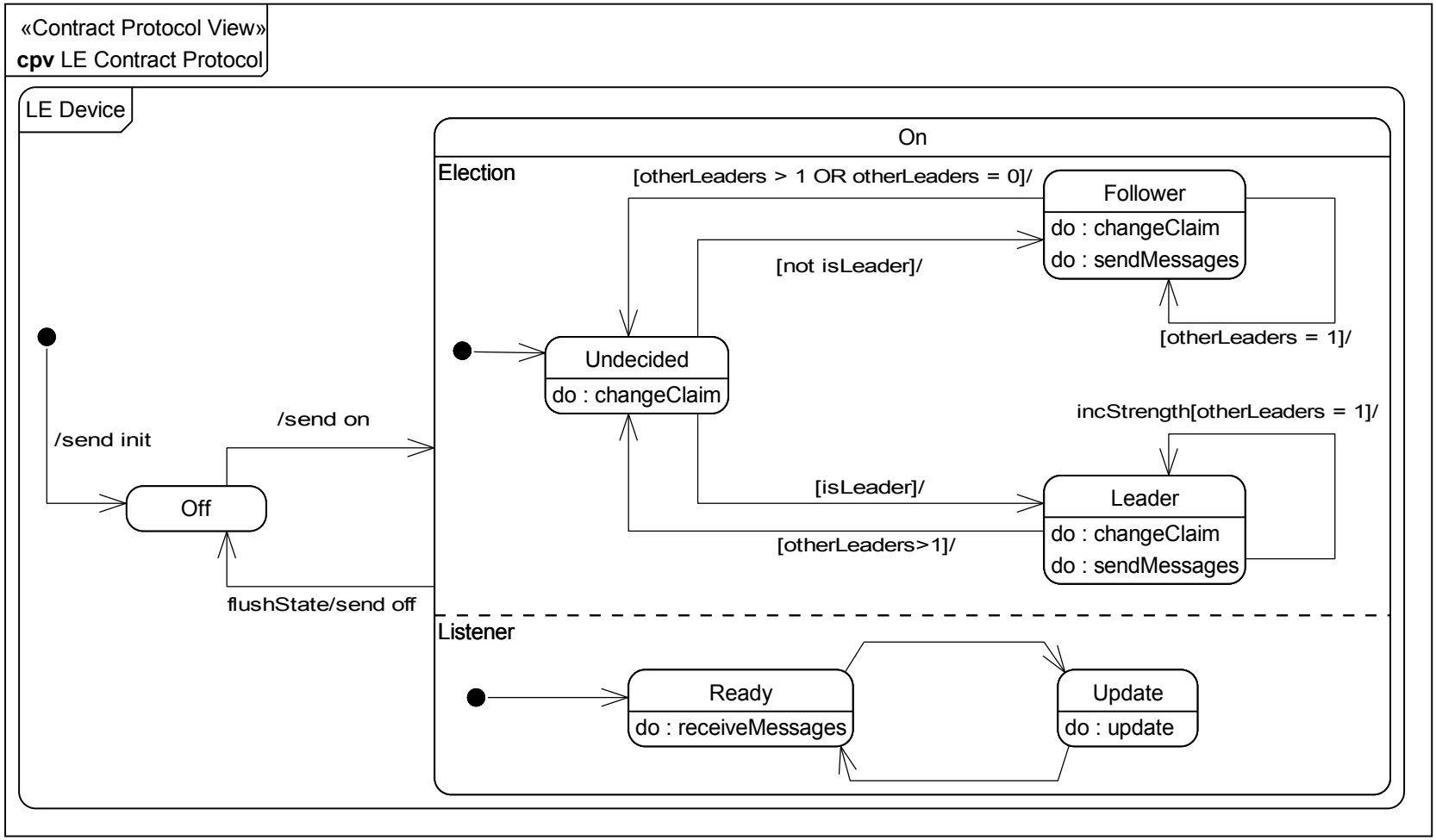
LE Device Contract Definition View



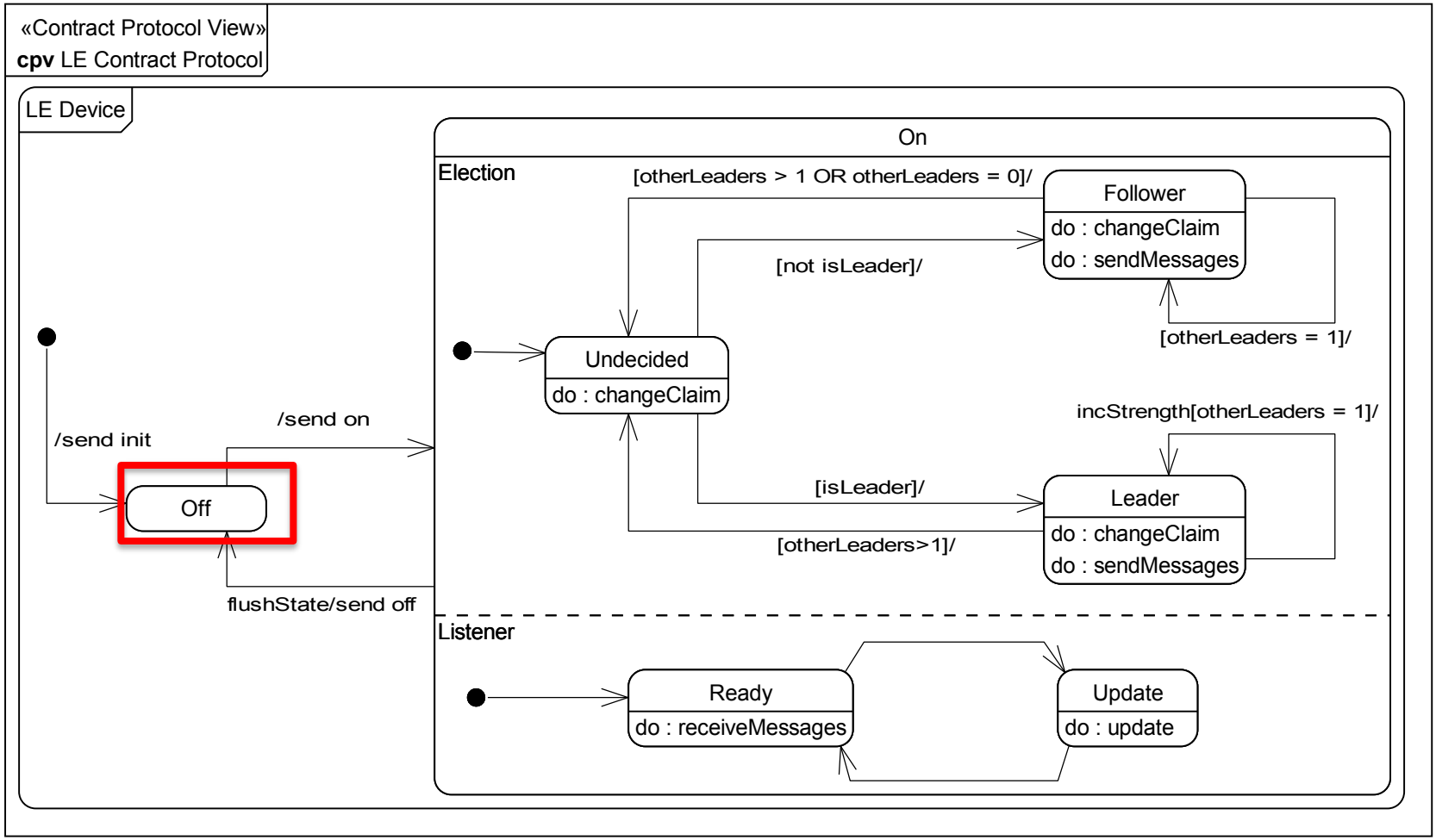
LE Device Contract Definition View



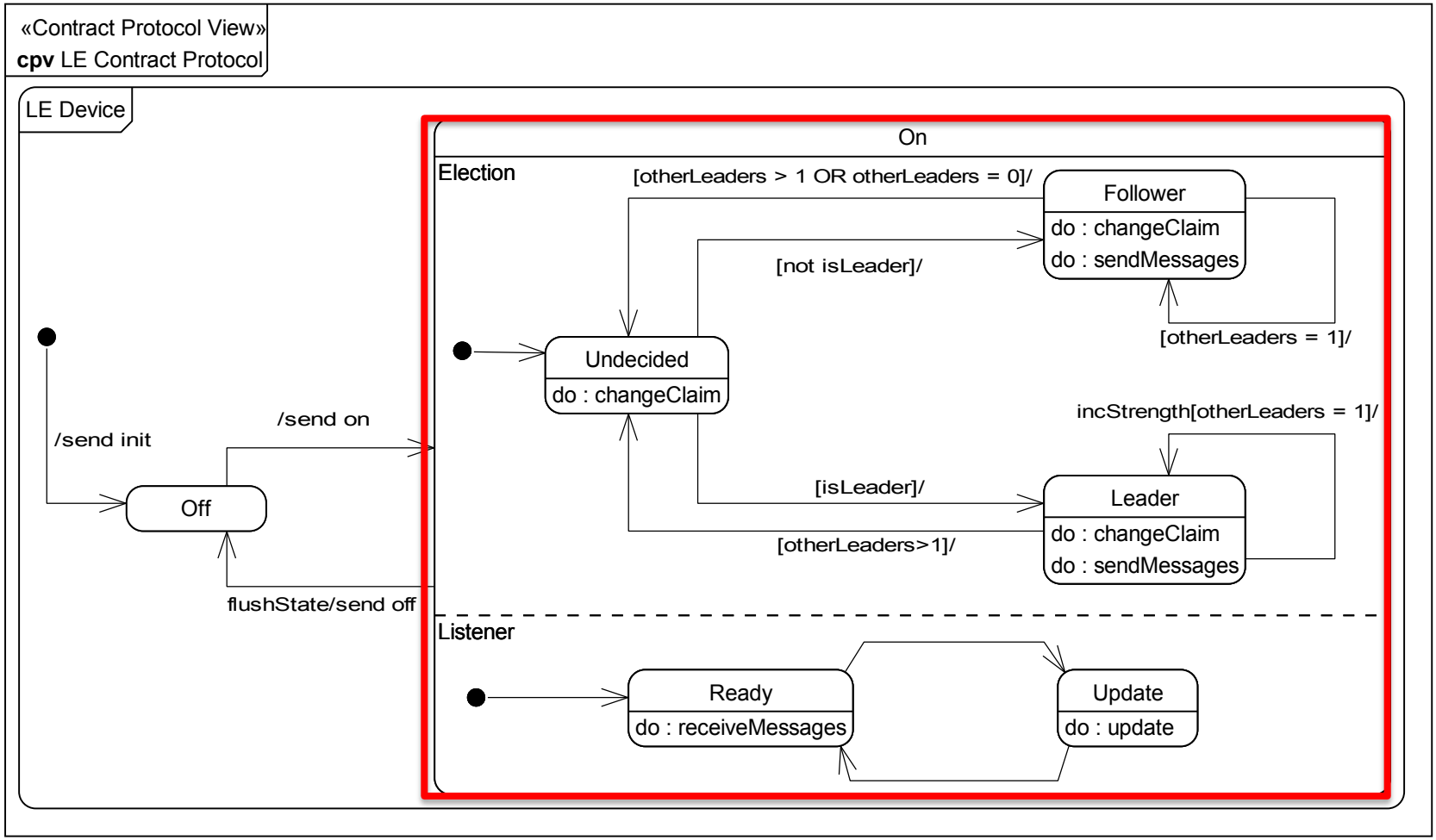
LE Device Contract Protocol View



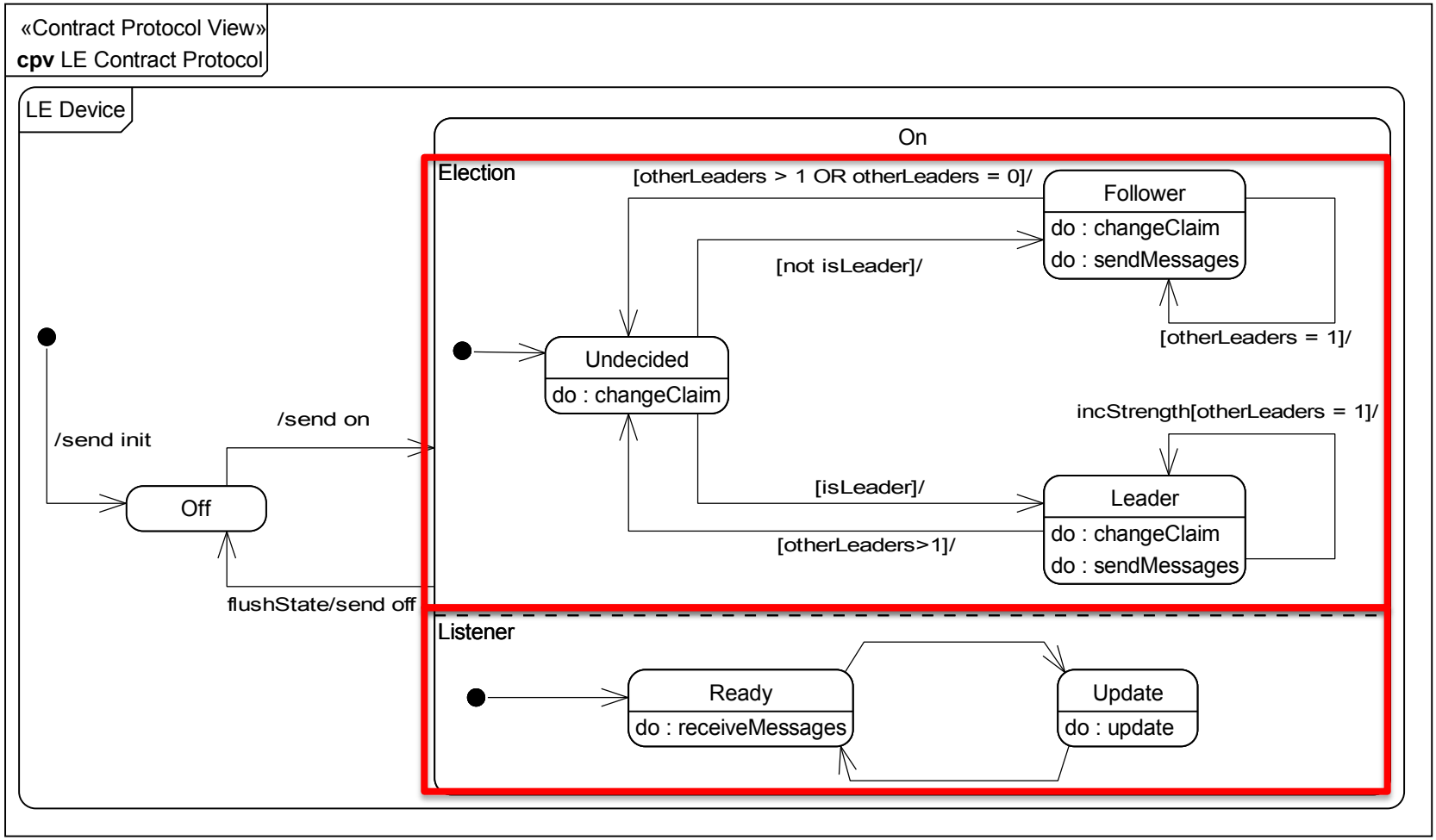
LE Device Contract Protocol View



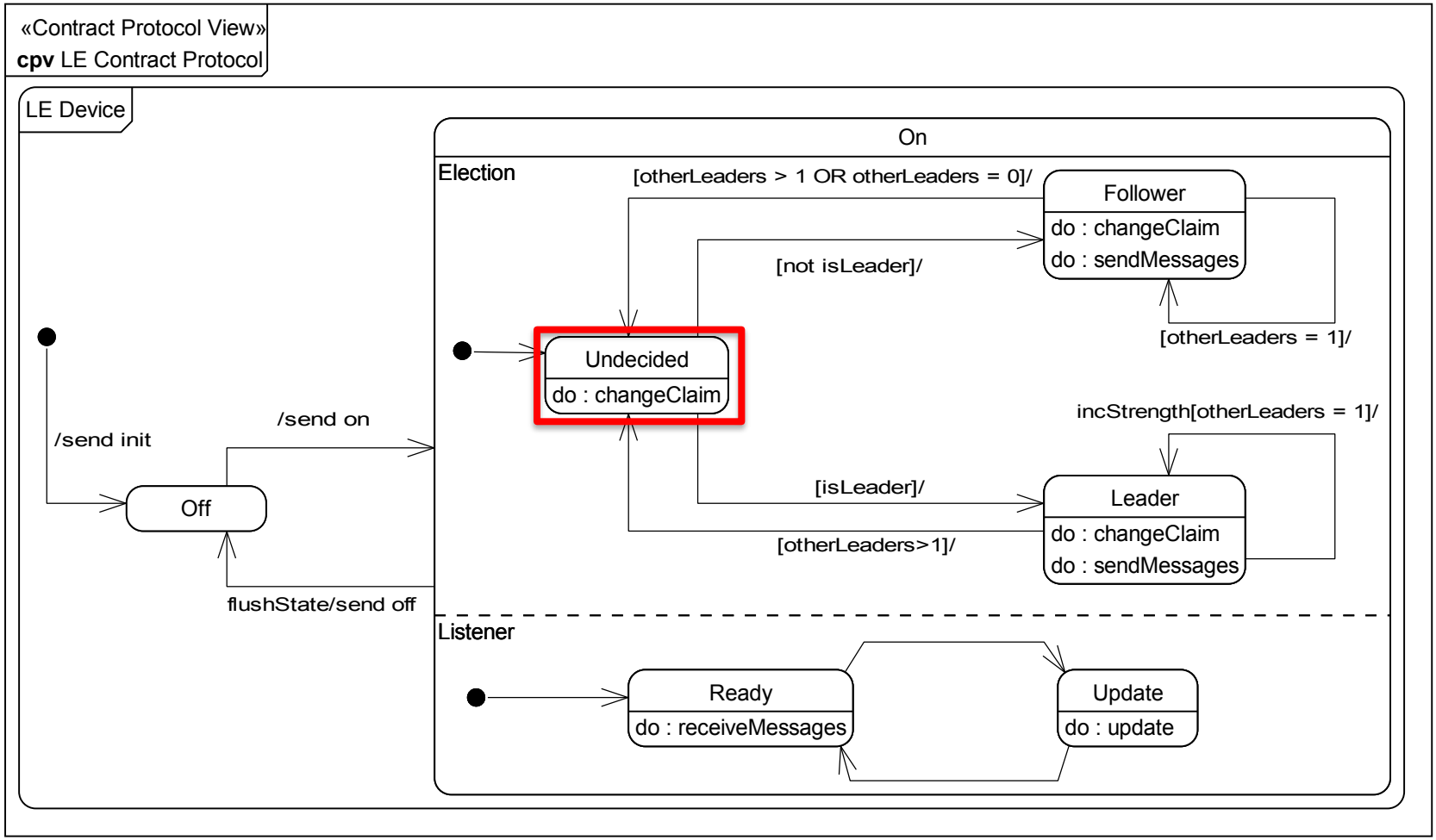
LE Device Contract Protocol View



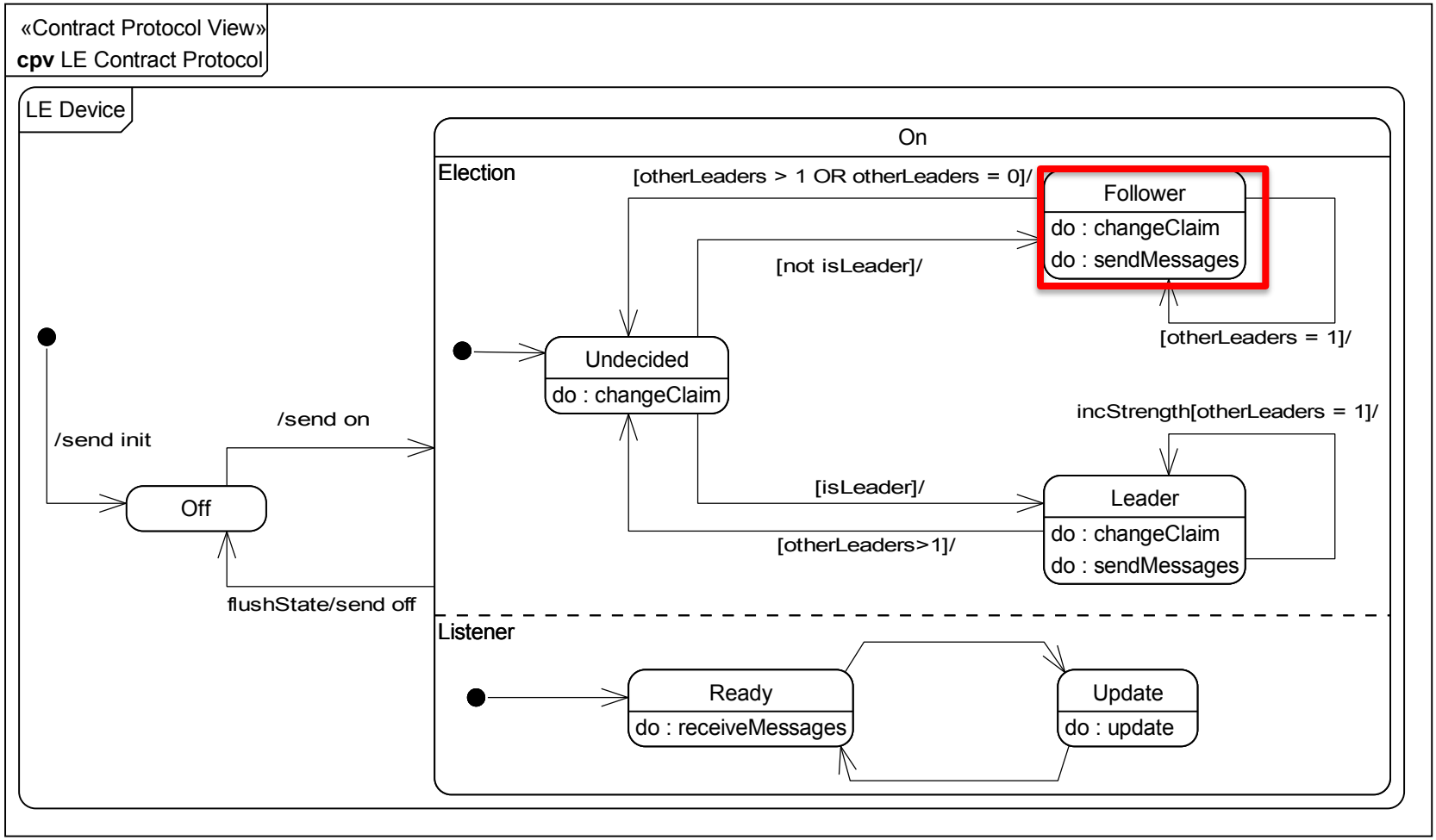
LE Device Contract Protocol View



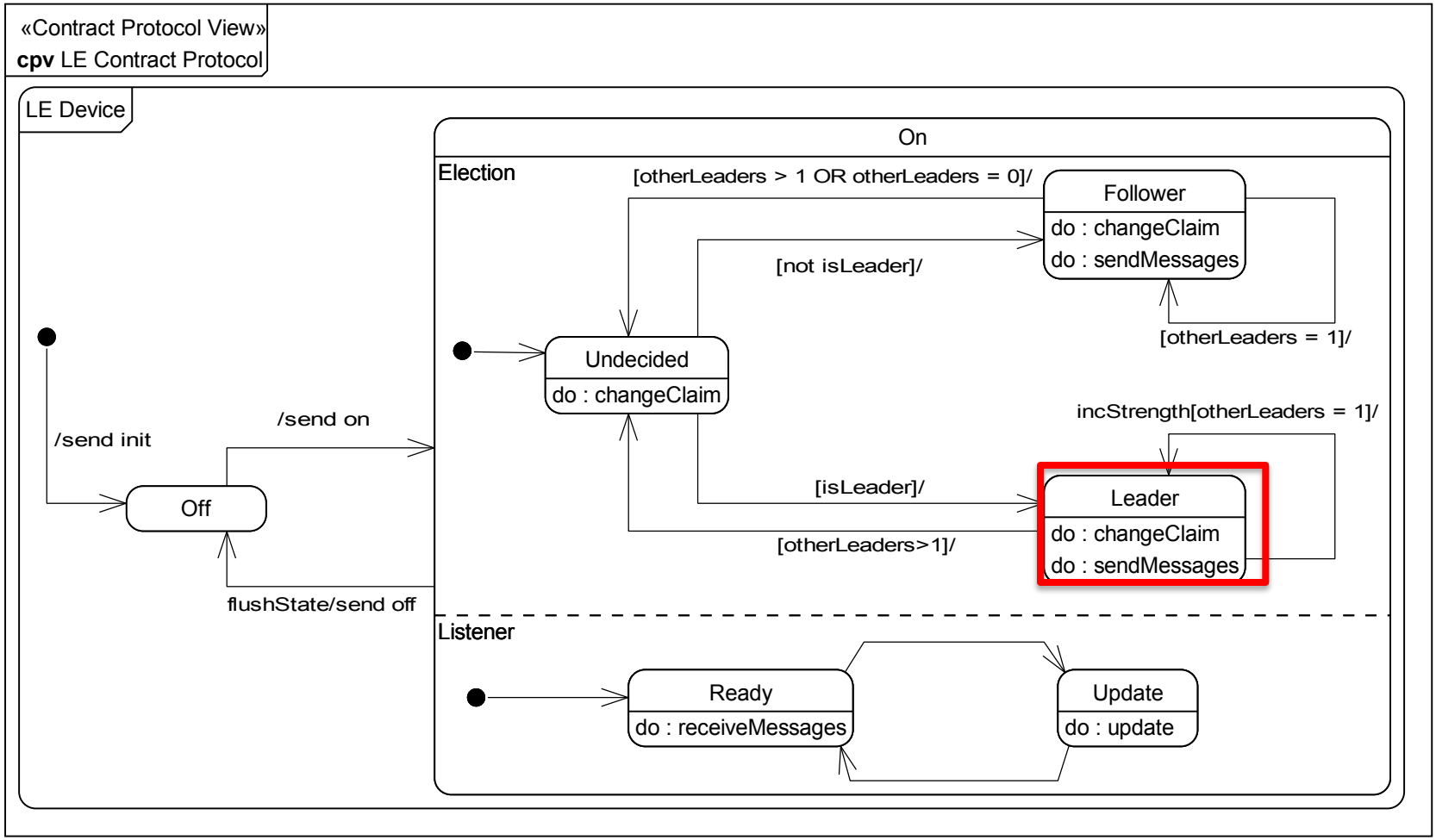
LE Device Contract Protocol View



LE Device Contract Protocol View



LE Device Contract Protocol View



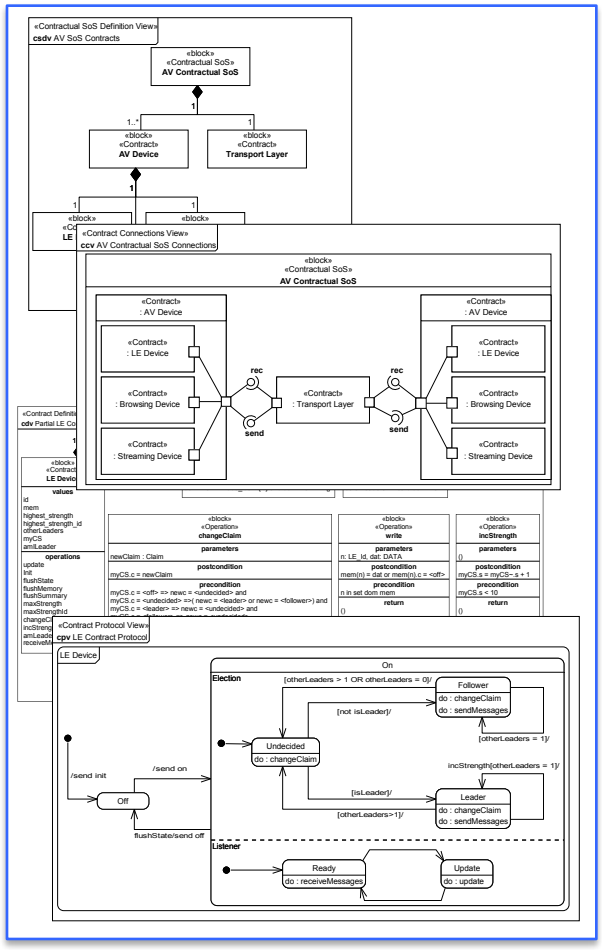
Overview

- Systems of Systems
- Case study
- Modelling
- **Analysis**
- Conclusions and future work

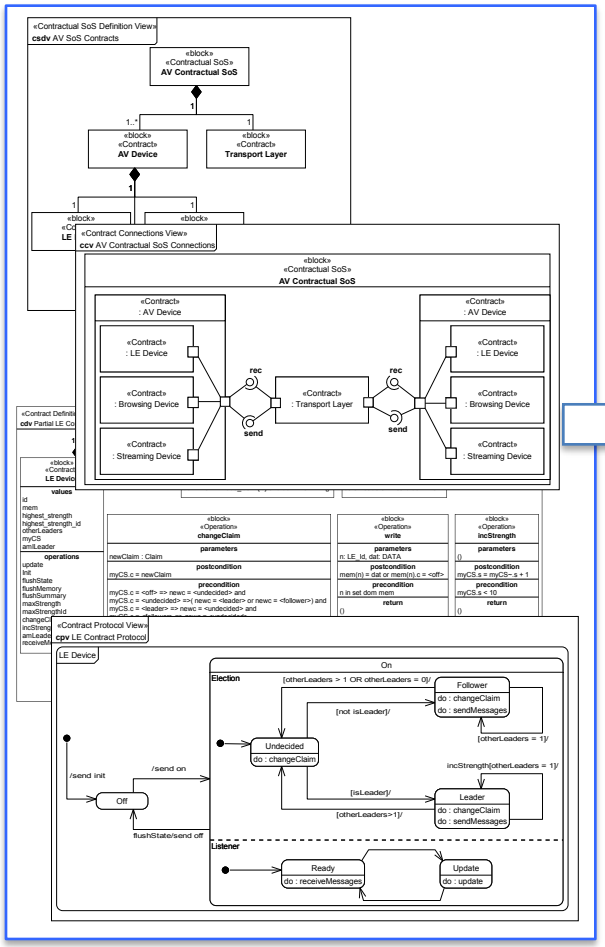
Model-based Analysis

- Translate SysML contract model to formal notation COMPASS Modelling Language (CML)
  - Contracts are defined in terms of communicating *processes*
  - Processes contain *datatypes*, *variables*, *operations* and *actions*
- Verify requirement of emergent behaviour using CML tool Symphony
  - Formal semantics allows range of formal analyses

Analysing the Model

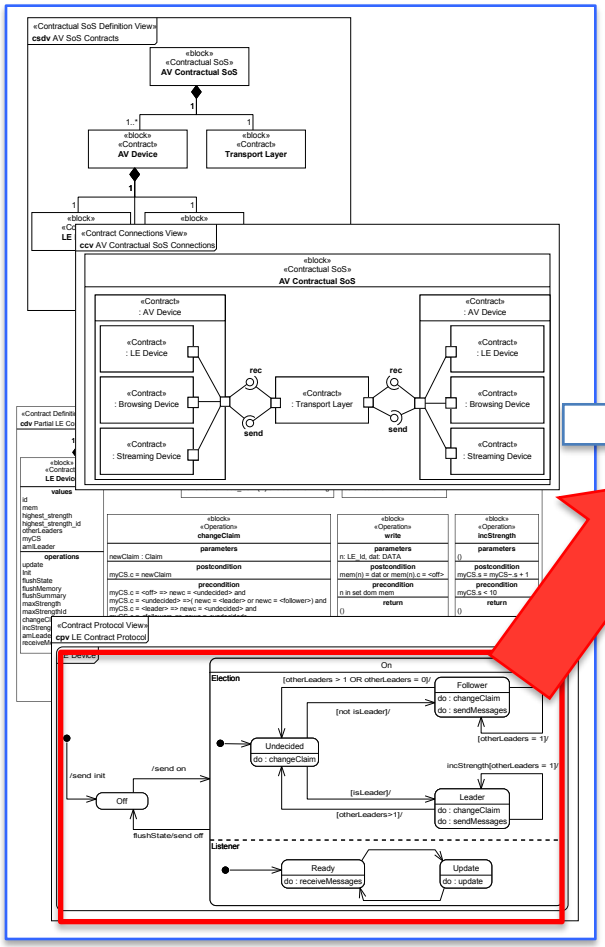


Analysing the Model



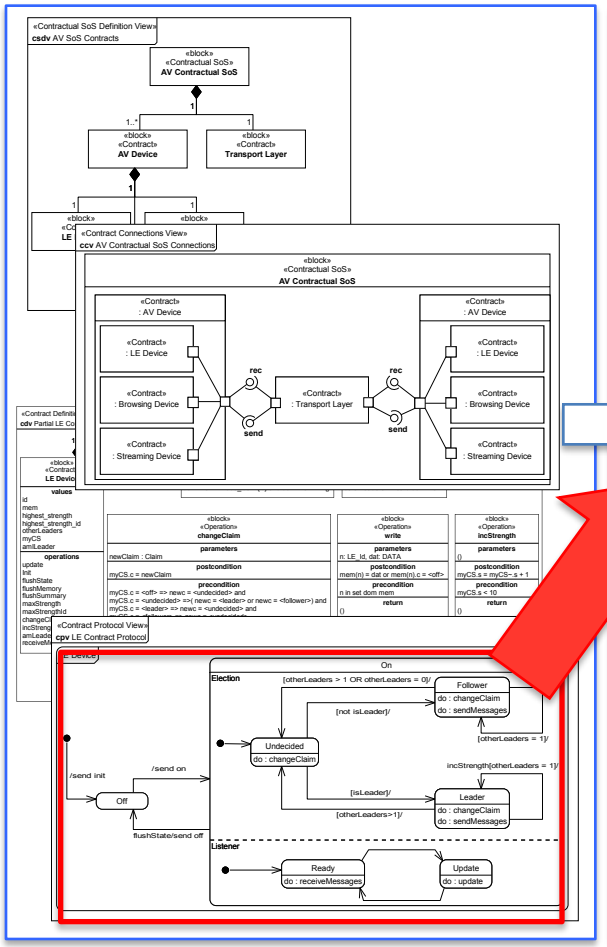

```
process LE_Device = i : nat @
begin
  …
  actions
    Off = on!id -> (Undecided /_\ off!id
      -> flushState();Off)
    Undecided = changeClaim(<undecided>);
      Listener;([isleader]& Leader
                []
                [not isleader]& Follower)
    Leader = …
    Follower = …
    Listener = …
end

process TransportLayer =
begin
  …
end

process AllLEDevices =
      || i in set le_ids @ (LE_Device(i))

process AVSoS= AllLEDevices
           [|{|interface_channels|}|]
             TransportLayer
```

Analysing the Model

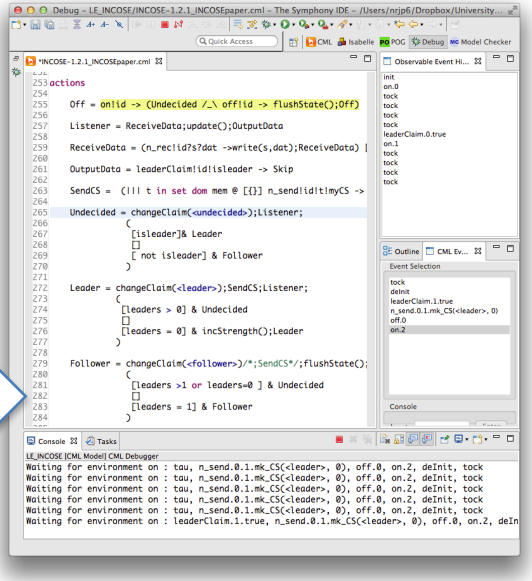



```
process LE_Device = i : nat @
begin
  …
  actions
    Off = on!id -> (Undecided /_\ off!id
      -> flushState();Off)
    Undecided = changeClaim(<undecided>);
      Listener;([isleader]& Leader
                []
                [not isleader]& Follower)
    Leader = …
    Follower = …
    Listener = …
end

process TransportLayer =
begin
  …
end

process AllLEDevices =
      || i in set le_ids @ (LE_Device(i))

process AVSoS= AllLEDevices
            [|{|interface_channels|}|]
              TransportLayer
```

Analysing the Model



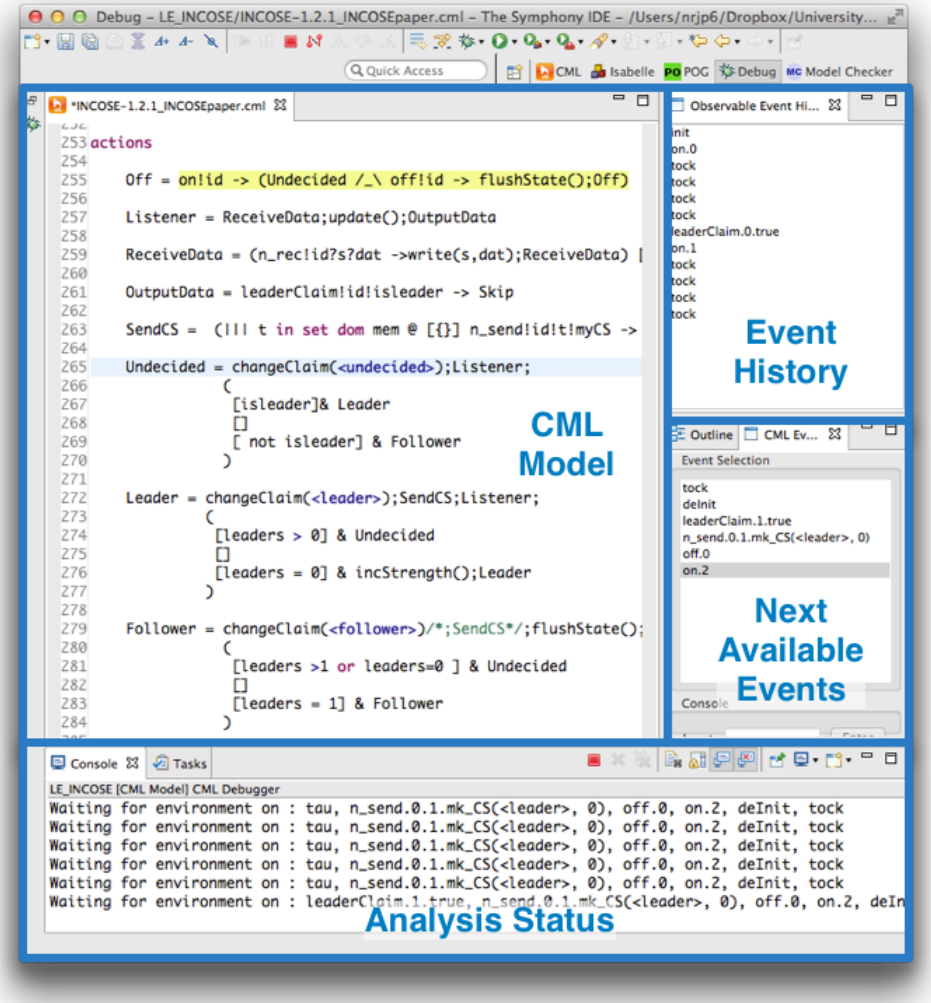

```
process LE_Device = i : nat @
begin
  …
  actions
    Off = on!id -> (Undecided /_\ off!id
      -> flushState();Off)
    Undecided = changeClaim(<undecided>);
      Listener;([isleader]& Leader
                []
                [not isleader]& Follower)
    Leader = …
    Follower = …
    Listener = …
end

process TransportLayer =
egin
 …
end

process AllLEDevices =
      || i in set le_ids @ (LE_Device(i))

process AVSoS= AllLEDevices
          [|{|interface_channels|}|]
            TransportLayer
```



Symphony tool

- Analyse leader election emergent behaviour
- Simulate execution of model
- Model checking

CML Model Simulation



- Used Symphony simulator to execute traces of CML model
- Model does not meet requirement R1.1
  - Can have more than one leader
  - However, quickly resolved
  - Incorrect model or incorrect requirement?
- New CSs may be added and emergent behaviour maintained

Overview

- Systems of Systems
- Case study
- Modelling
- Analysis
- **Conclusions and future work**

Conclusions

- Established need for contractual definition of constituent systems
- Defined and demonstrated contracts pattern with industrial proof of concept study
  - Using SysML and CML
- Demonstrated analysis of CS contracts to ensure required emergence is maintained
  - Simulation of CML model
  - Resulting in clarification of requirements

Future Work

- Integrate SoS engineering frameworks
  - e.g. fault modelling and analysis, testing
- Enhance contract pattern
  - non-functional properties and security features
- Modelling SoS-level contracts in pattern
- Consequences of contract composition
- Automated contract conformance

Future Work: Modelling Power in Contracts

- In CPLab at Newcastle University, working in SoS and Cyber-Physical Systems
- Existing pattern allows only representation of digital phenomena
- In CPS modelling, need to represent physical properties (e.g. power)
- Initial results modelling Smart Grids with contracts and co-modelling

- research.ncl.ac.uk/cplab

jeremy.bryans@ncl.ac.uk @JWBryans
john.fitzgerald@ncl.ac.uk @NclFitz
richard.payne@ncl.ac.uk @riffio
KRT@bang-olufsen.dk

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

COMPASS

thecompassclub.org