DANSE
Designing for Adaptability and Evolution in Systems of Systems Engineering

Workshop on Achievements in
Systems of Systems Research and Innovation
28th of May, 2015

Valerio Senni
Advanced Laboratory on Embedded Systems
United Technologies Systems & Controls Engineering Group

Advanced Laboratory on Embedded Systems
A Research and Innovation Company

United Technologies | **SYSTEMS & CONTROLS ENGINEERING**

UTC BUSINESSES (2014)

Commercial

2014 net sales $65.1 billion

Aerospace

OTIS

United Technologies
Climate | Controls | Security

PRATT & WHITNEY
DEPENDABLE ENGINES

Sikorsky

UTC Aerospace Systems

United Technologies | SYSTEMS & CONTROLS ENGINEERING

DANSE EU PROJECT FACTS

- Period 2011-2015 (39+3)
- Budget 11.6 M
- Consortium

- Main Objectives
  - Develop a comprehensive methodology for SoS Engineering
  - Validate methodology on four industrial test-cases
- Test Cases
  - Fire Emergency Response (AIRBUS)
  - Integrated Water Treatment and Supply (IAI)
  - Air Traffic Management (THALES)
  - Automated Ground Transport (CARMEQ)

SYSTEMS OF SYSTEMS

Composed out of complex, **autonomous** and **heterogeneous** sub-systems.
Subject to **dynamicity** and change.

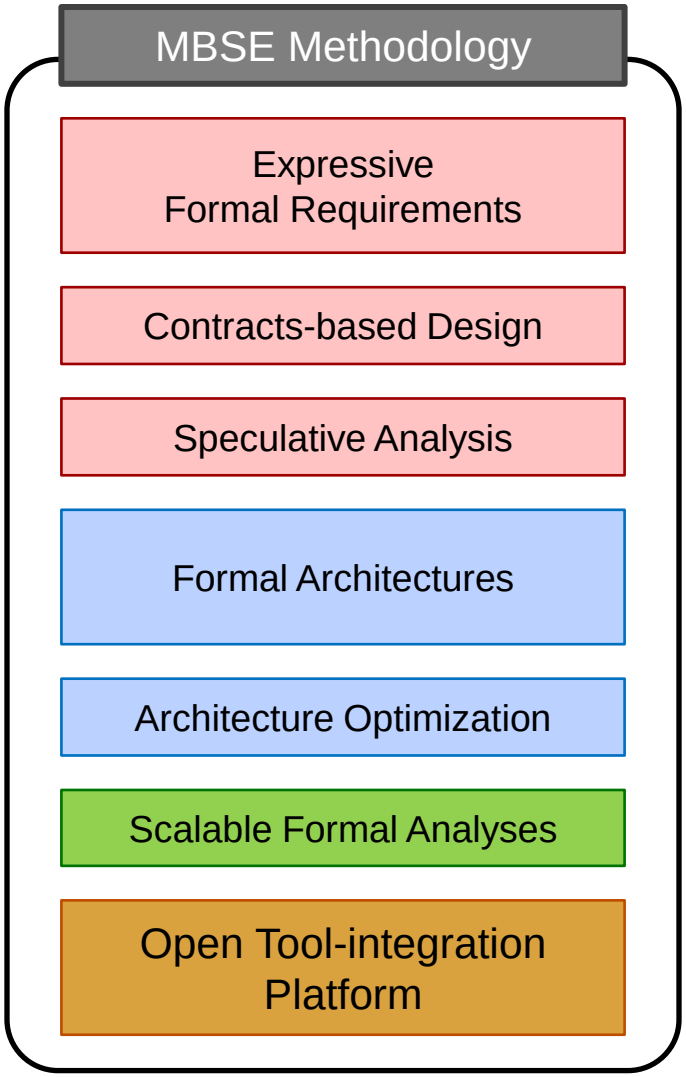
Big challenges stemming from the risk of **emergent and unpredicted behaviors**, that should be handled dynamically and with **evolutionary** approaches.

Need to preserve achievement of both SoS and CSs goals, by **resolving conflicts**.

A Methodology to Address SoS Challenges

SoS Challenges

- Operational independency
- Heterogeneous Systems Integration
- Constituent Systems cooperation
- Design process complexity
- Reliability and Safety
- Robustness to dynamicity
- Capability of evolution
- Identification of Emergent Behaviors and capability to react to them
- …



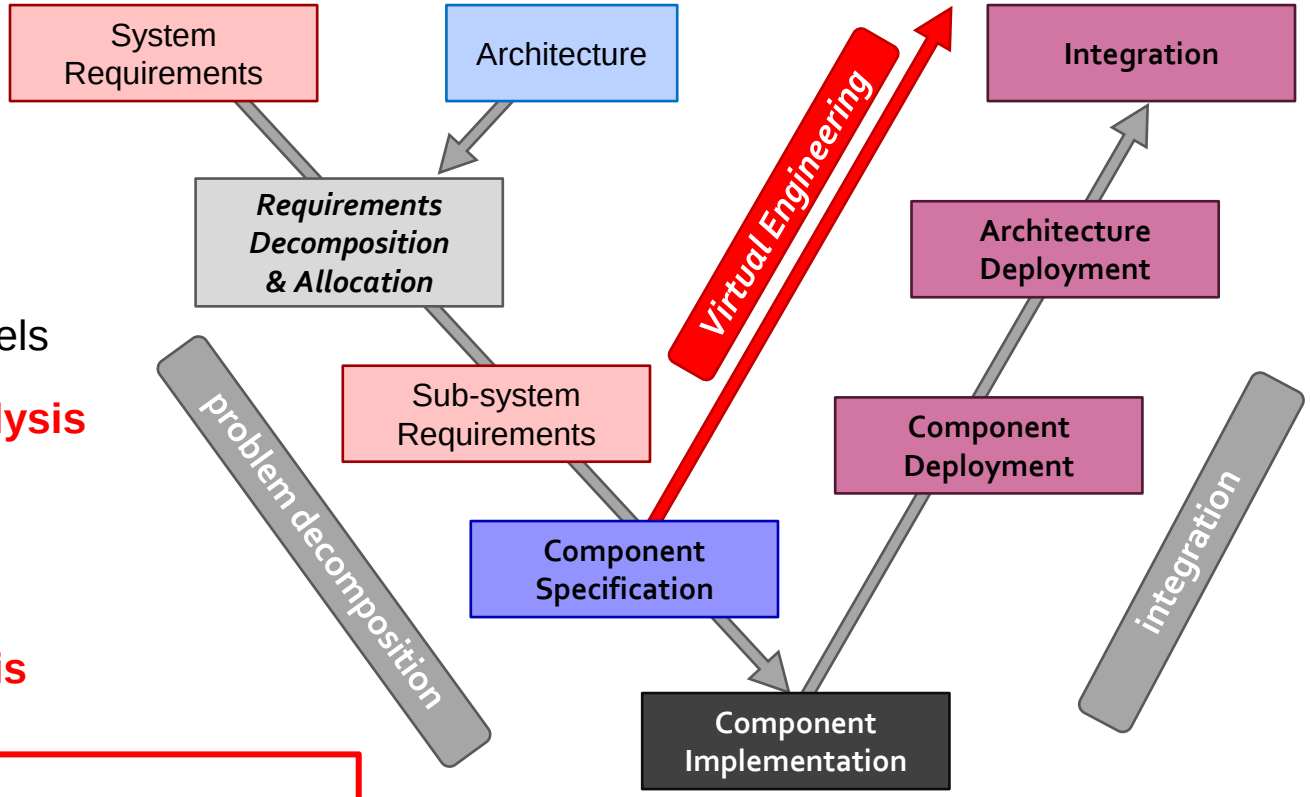
A Methodology to Address SoS Challenges

SoS Challenges

- Operational independency
- **Heterogeneous Systems Integration**
- **Constituent Systems cooperation**
- **Design process complexity**
- Reliability and Safety
- Robustness to dynamicity
- **Capability of evolution**
- Identification of Emergent Behaviors and capability to react to them
- …

MBSE Methodology for SoS

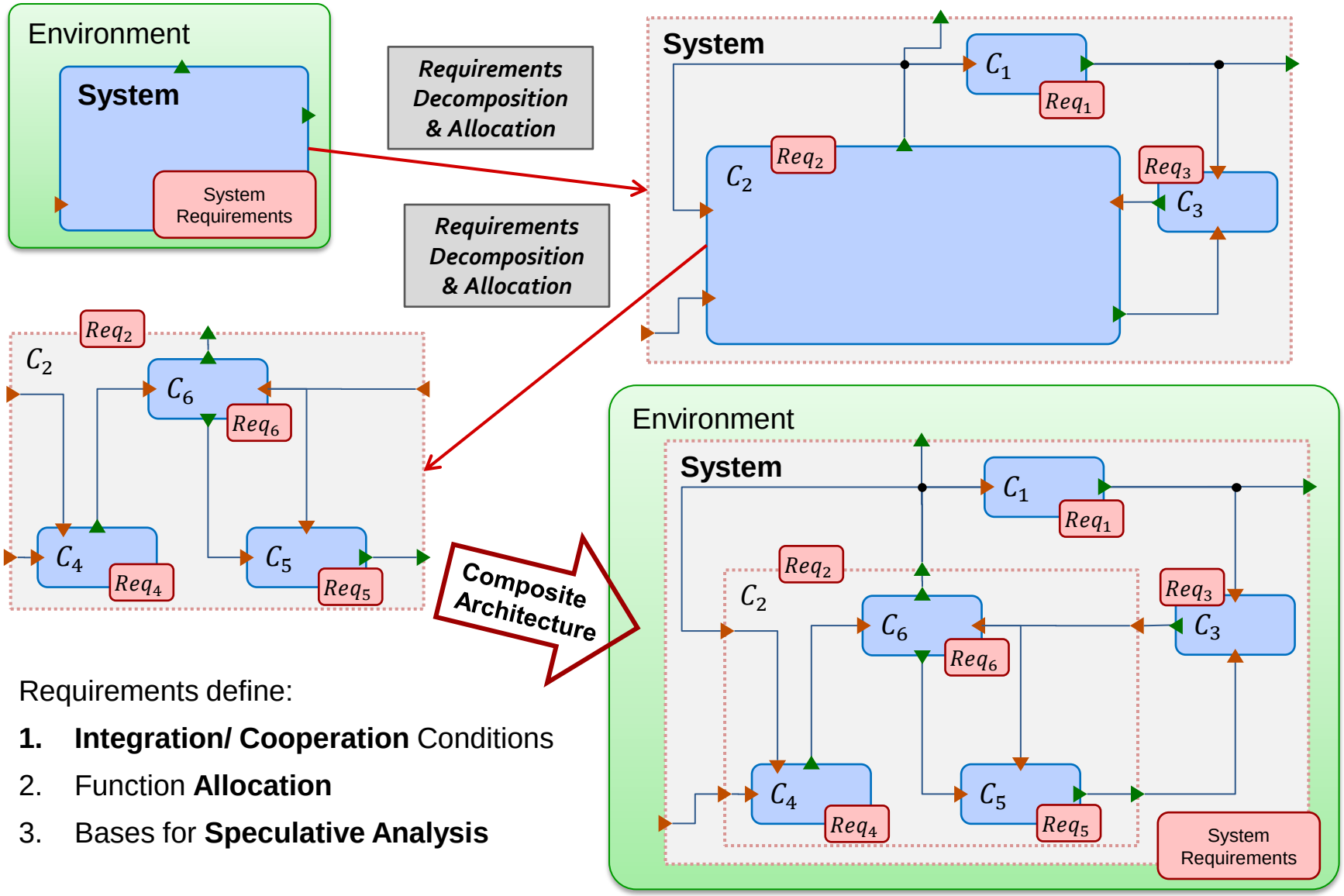
1. **Formal Artifacts**
   a. Requirements
   b. Architecture
   c. Behavioral Models
2. **Design-time Analysis**
3. Implementation
4. Integration
5. **Run-time Analysis**
6. Change need



Model-Based approaches
enable early application of analyses
to anticipate the identification of errors
and avoid expensive deployment-time errors

Iterative, to support evolution

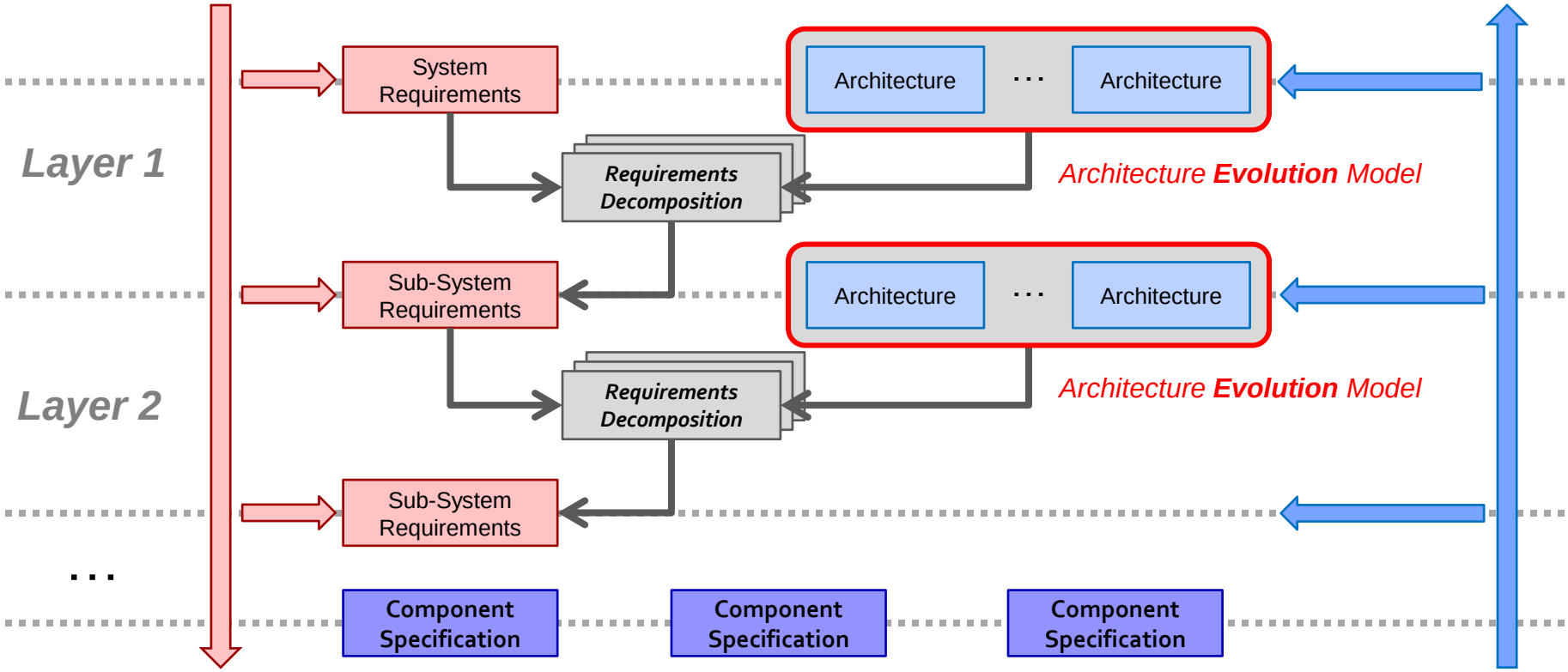
ARCHITECTURE-DRIVEN REQ.S DECOMPOSITION



Requirements define:

1. **Integration/ Cooperation** Conditions
2. Function **Allocation**
3. Bases for **Speculative Analysis**

ARCHITECTURE DRIVEN METHODOLOGY

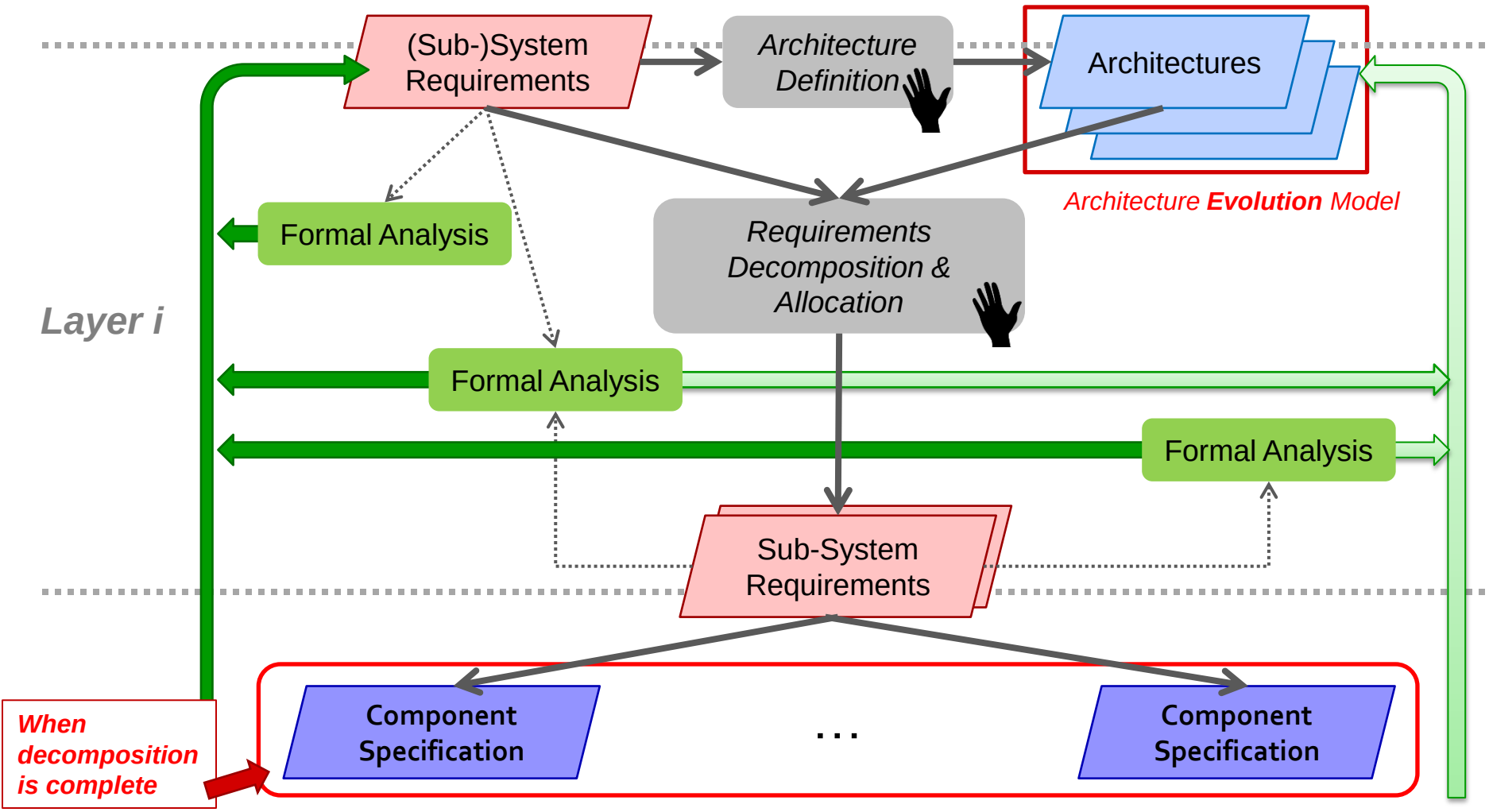


Requirements Refinement

Architecture Abstraction

1. Design **complexity** mastered through decomposition
2. Architecture evolution models capture **evolution capabilities**
3. Requirements and Architecture improved using **feedback** from lower design levels

Decomposition and Role of Formal Analysis

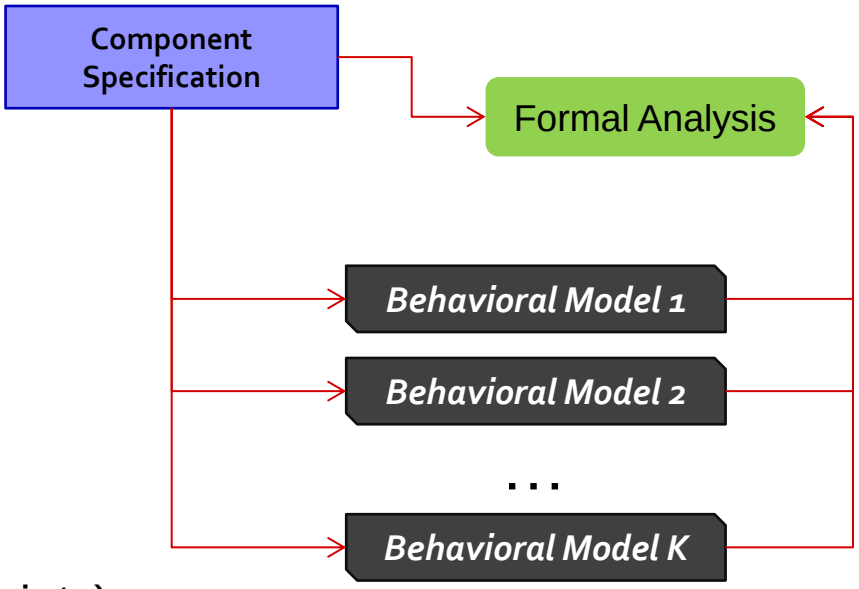


Requirements feedback

Architecture feedbacks

Formal Analysis provides valuable feedback during design

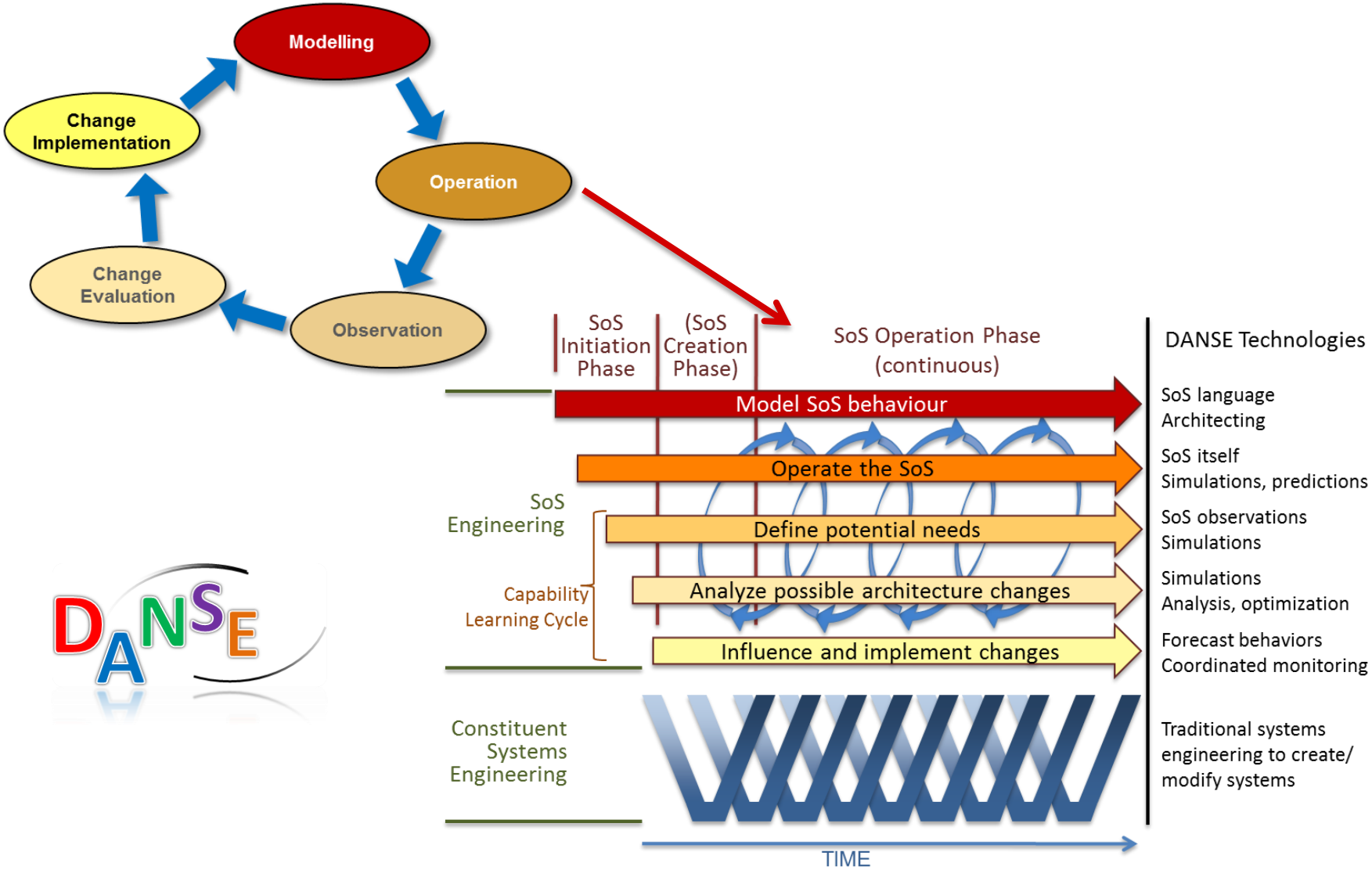
DESIGNING COMPONENTS BEHAVIOR



At Component Specification level:

- No remaining decisions on Architecture
- Need to provide a **model of behavior**
- Several possible models:
  - Different **abstraction level**
  - Efficiency/Quality
- **Stochastic** Models (Variability vs Uncertainty)
  - Abstraction to avoid full decision on part of the behavior (within Designer's decision cone)
  - Abstraction to capture Uncertainties (out of Designer's decision cone) ← Robust Design
- Formal Analysis role:
  local assessment against (formal) Specifications entails composite system correctness against **global** Requirements ← Speculative Analysis

DANSE MBSE METHODOLOGY FOR SOS

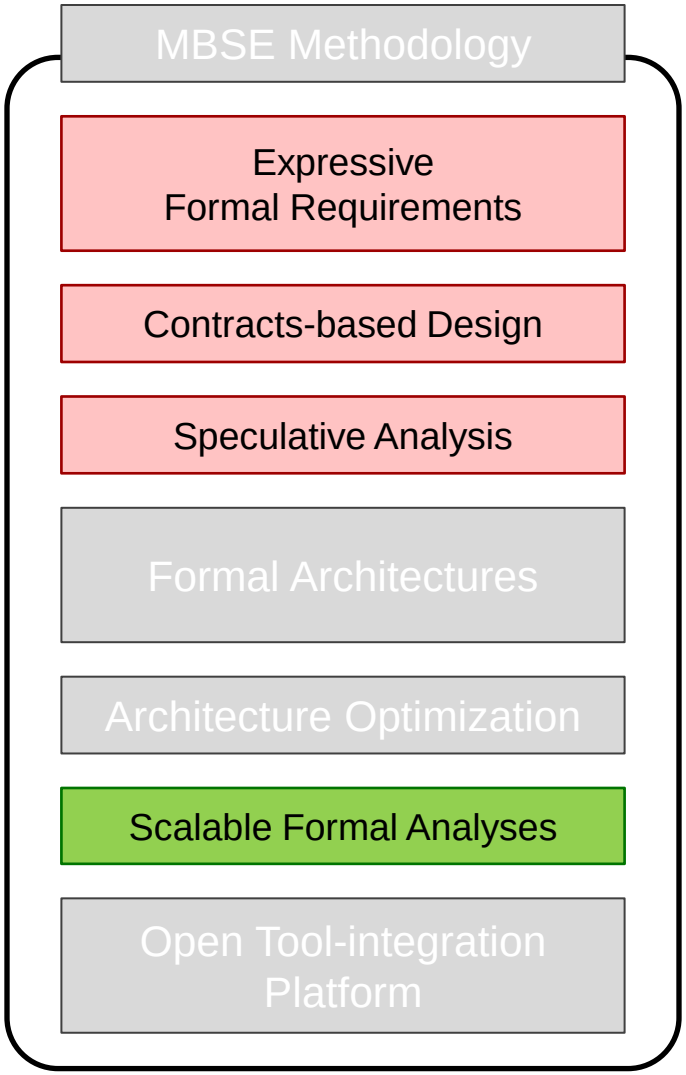


FORMAL REQUIREMENTS

A Methodology to Address SoS Challenges

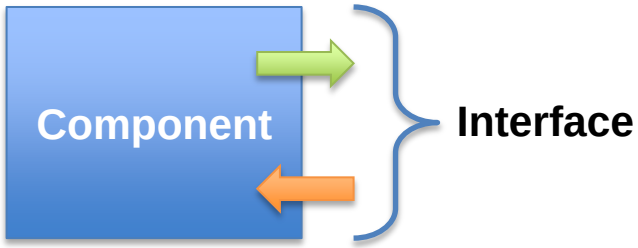
SoS Challenges

- Operational independency
- **Heterogeneous Systems Integration**
- **Constituent Systems cooperation**
- Design process complexity
- **Reliability and Safety**
- Robustness to dynamicity
- Capability of evolution
- **Identification of Emergent Behaviors and capability to react to them**
- …



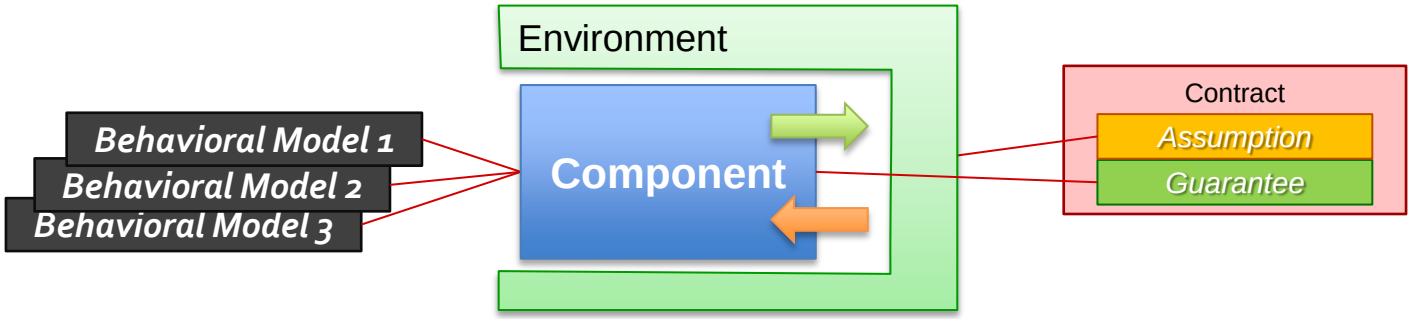
ASSUME/GUARANTEE CONTRACTS

- Contracts are ***"Strong"*** **formal requirements**

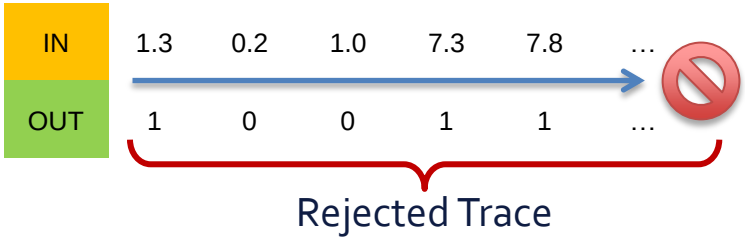


- They specify a component at the **interface** level by:
  - Assumptions over **inputs** (Environment)
  - Guarantees over **outputs** (Implementation)

Contract: Assumption ⇒ Guarantee (if applicable, it ensures…)



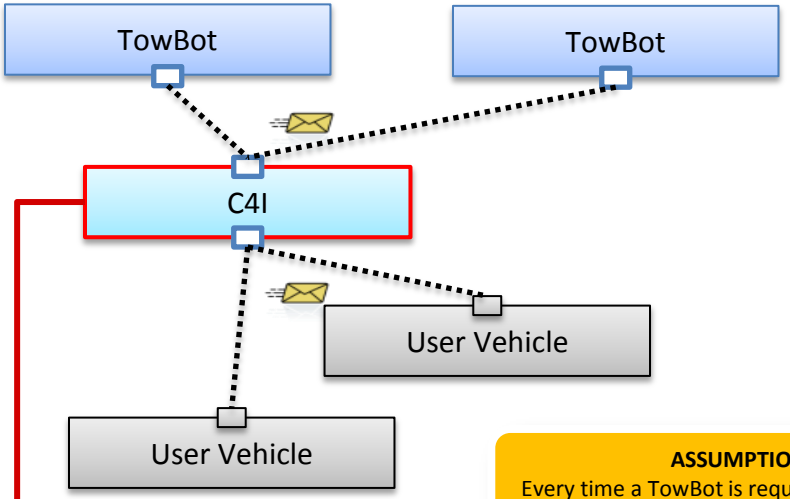
- Contracts are **more than types**: they are behavioral descriptions



CASE STUDY: THE ATS SYSTEM (SPRINT)

ATS: Automatic Towing Service

- Vehicles request towing services
- ATS Components
  - Centralized control subsystem (C4I)
  - User vehicles send requests to C4I
  - Tow-bot receives commands from the C4I
- Wireless communication between C4I, user vehicles and tow-bots
- Asynchronous execution of the components

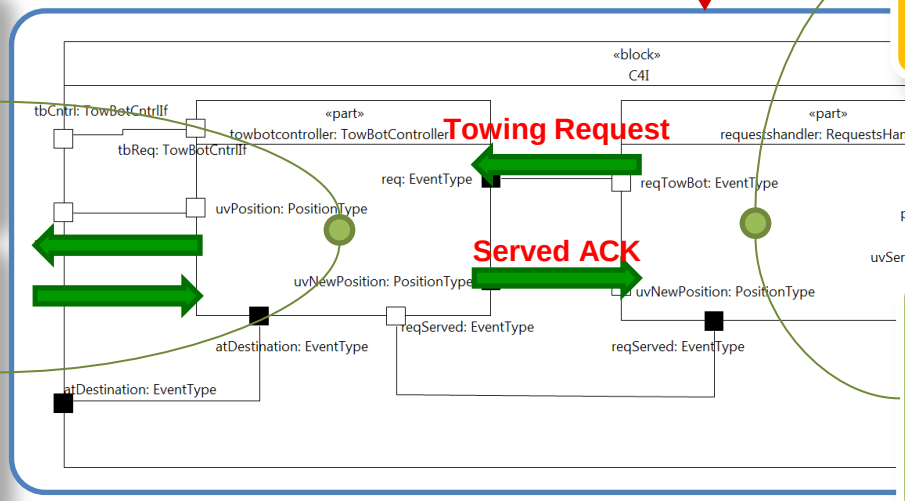


ASSUMPTION
Every time a TowBot is requested , a TowBot dispach confirmation follows

Everytime [reqTowBot] **then** [reqServed] **follows**

ASSUMPTION
Every time a TowBot dispatch request is send, the TowBot arrives at destination

Everytime [tbReq] **then** [atDestination] **eventually**



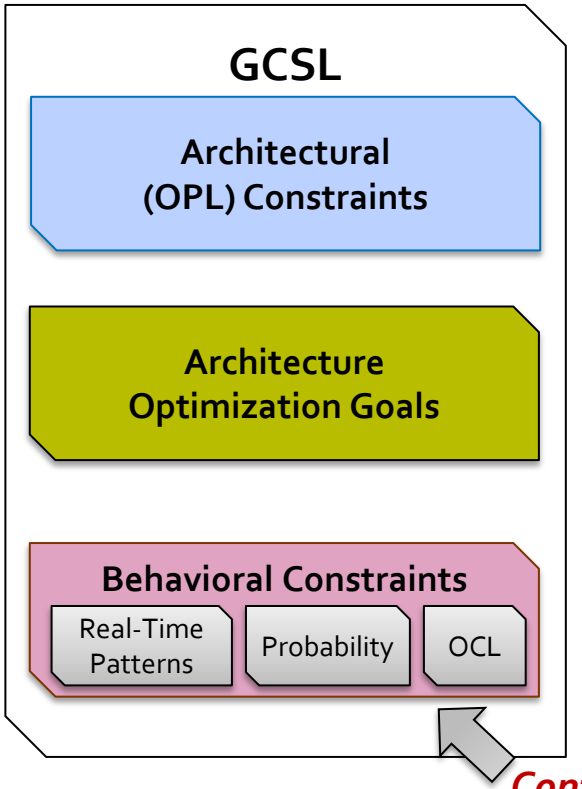
GUARANTEE
Every time the TowBot controller receives a dispatch requests, it sends a request to the TowBot in 5ms

Everytime [req] **then** [tbReq] **happens within 5ms**

GUARANTEE
Every time a user vehicle requests a TowBot, it will be served in 1hour

Everytime [uvReq] **occurs implies** [uvServed] **within 1hour**

GCSL: A SoS-specific Formal Requirements Language



Requirements on # of components, # of connections, weight, cost, length, …

```
physLink[g] >=
    funcLink2physVLink12[f][g] +
    funcLink2physVLink21[f][g] ;
```

Objective functions on architecture cost parameters

$$\text{cost}(i) = \sum_{j \in \textbf{Scope}(\text{Cost}) \textbf{ inter Linked}\textbf{Scope}(\text{Null, isContainedIn},i).keySet()} \text{isSelected}(j) \cdot \text{Cost}(j) \quad \forall\, i \in \textbf{Scope}(\text{totalCost})$$

SoS and CSs (abstract) behavior description

```
SoS.itsFireStations->forAll(fireStation |
  whenever [
    fireStation.hostedFireFightingCars->forAll(isAtFireStation = false)
  ] occurs [
    fireStation.hostedFireFightingCars->exists(isAtFireStation = true)
  ] occurs within [3 hours]
)
with probability 0.87
```

Contracts specification language

Architectural Constraints
- **Strong** Requirements
- Satisfied at Architecture-design time

Behavioral Constraints
- **Strong** Requirements
- Satisfied at Behavior-design time

Optimization Goals
- **Weak** Requirements
- Maximized at Architecture-optimization time

GCSL provides a multi-view specification language

GCSL Behavioral - Features

- ✓ Real-Time, Numeric Constraints, Probability

  within 3.1 hours, $Component_1.port_3 \geq Component_2.port_1 + 3$, with probability 0.99

- ✓ Natural Language Pattern -based (safety & liveness patterns)

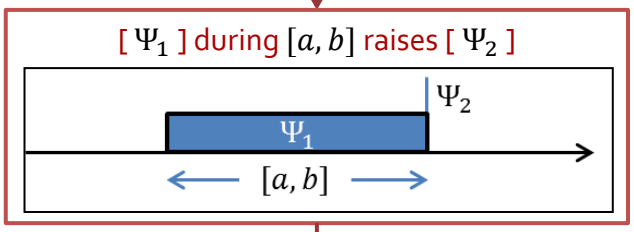
```
1   [ OCL-prop ] implies [ OCL-prop ] holds forever |
2   always [ OCL-prop ] |
3   whenever [ OCL-prop ] occurs [ OCL-prop ] holds |
4   [ OCL-prop ] implies [ OCL-prop ] during following interval |
5   [ OCL-prop ] during interval raises [ OCL-prop ] |
6   [ OCL-prop ] during interval implies [ OCL-prop ] during interval then [ OCL-prop ] during interval |
7   [ OCL-prop ] occurs int times during interval raises [ OCL-prop ] |
8   [ OCL-prop ] occurs at most int times during interval |
9   whenever [ OCL-prop ] occurs [ OCL-prop ] holds during following interval |
10  whenever [ OCL-prop ] occurs [ OCL-prop ] implies [ OCL-prop ] during following interval |
11  whenever [ OCL-prop ] occurs [ OCL-prop ] does not occur during following interval |
12  whenever [ OCL-prop ] occurs [ OCL-prop ] occurs within interval
13  always during interval [ OCL-prop ] has been true at least [ OCL-expr ] % of time
14  at the end of interval [ OCL-prop ] has been true at least [ OCL-expr ] % of time
```

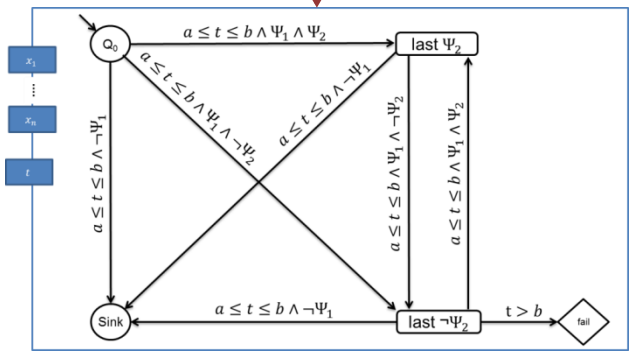
Real-time intervals

- ✓ Following the Assume-Guarantee paradigm
  - Assumptions on input **traces**
  - Guarantees on output **traces**

GCSL Behavioral – Patterns Semantics

User support through visual semantics





Patterns are translated to formal executable models (extended timed automata)

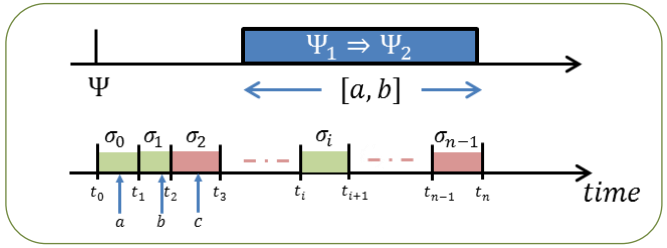
```
1   [ OCL-prop ] implies [ OCL-prop ] holds forever |
2   always [ OCL-prop ] |
3   whenever [ OCL-prop ] occurs [ OCL-prop ] holds |
4   [ OCL-prop ] implies [ OCL-prop ] during following interval |
5   [ OCL-prop ] during interval raises [ OCL-prop ] |
6   [ OCL-prop ] during interval implies [ OCL-prop ] during interval then [ OCL-prop ] during interval |
7   [ OCL-prop ] occurs int times during interval raises [ OCL-prop ] |
8   [ OCL-prop ] occurs at most int times during interval |
9   whenever [ OCL-prop ] occurs [ OCL-prop ] holds during following interval |
10  whenever [ OCL-prop ] occurs [ OCL-prop ] implies [ OCL-prop ] during following interval |
11  whenever [ OCL-prop ] occurs [ OCL-prop ] does not occur during following interval |
12  whenever [ OCL-prop ] occurs [ OCL-prop ] occurs within interval
13  always during interval [ OCL-prop ] has been true at leas
14  at the end of interval [ OCL-prop ] has been true at least
```

Natural-language based specification patterns

whenever [fireArea > 0] **occurs**
[FireFightingCar.isAtFire] **occurs**
within [1 *hour*]

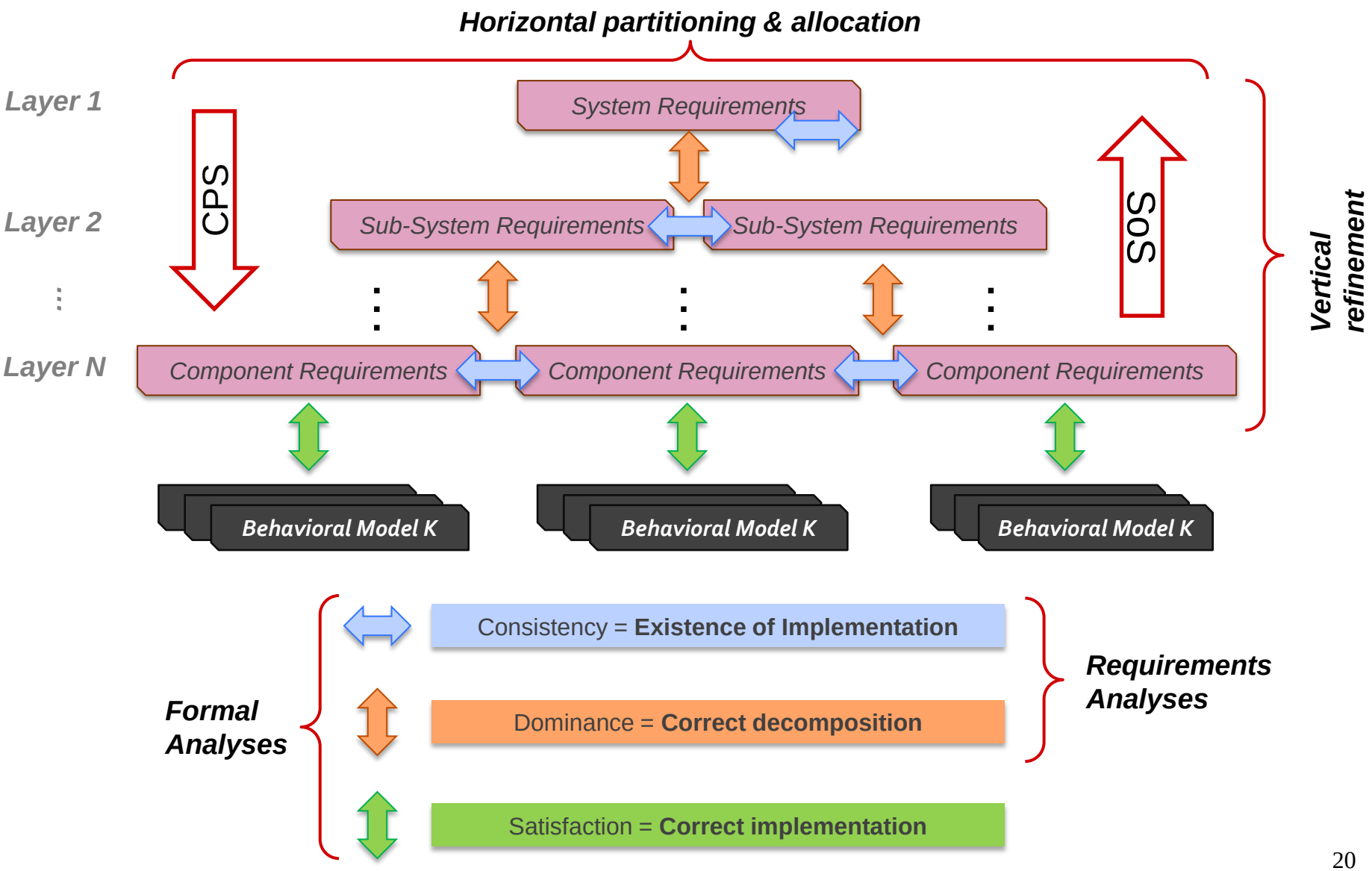
GCSL

Usable formal requirements



With underlying formalized semantics

FORMAL ANALYSES ON BEHAVIOURAL REQUIREMENTS



FORMAL REQUIREMENTS SUMMARY

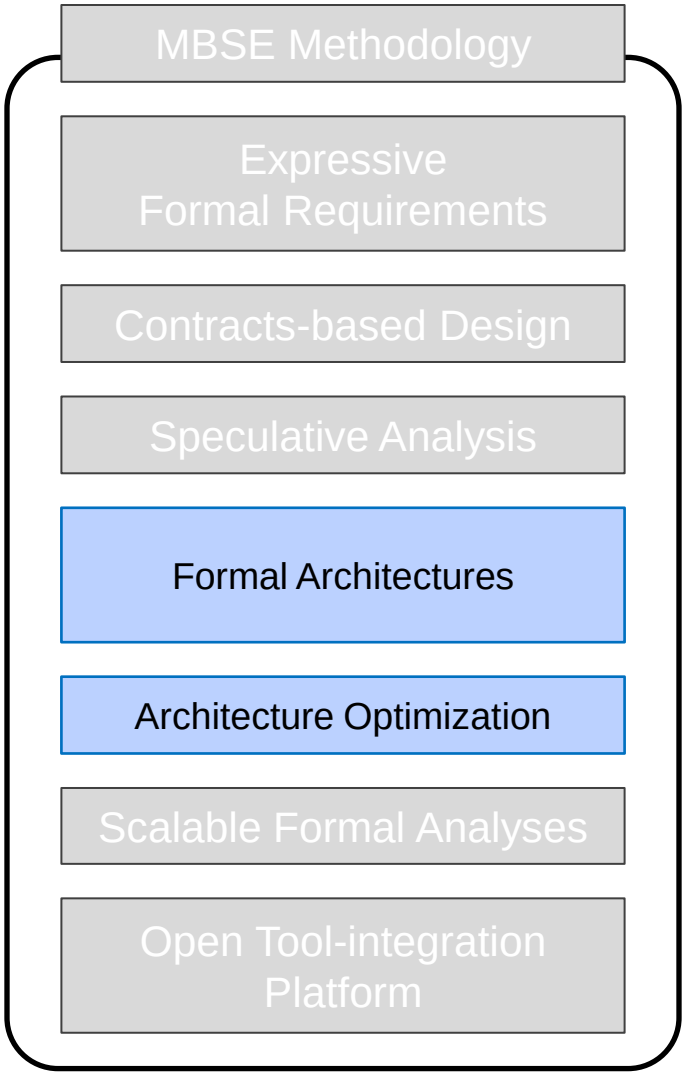
- Advantages of using Formalized Requirements
  - Automate **validation** and quality assessment
  - Adoption of Contract-based requirements enables to:
    support **decomposition**, function **allocation**,
    and **speculative (compositional) analysis**
  - Ensure semantic (i.e. more robust) **integration**
- "**Strong**" Requirements (Constraints)
  - Violations of Must Requirements shows **incorrect System design**
  - Require strict satisfaction
- "**Weak**" Requirements (Goals)
  - Capture **objective functions** for System tuning
  - Express **performance or cost** metrics
  - Require optimal System design

FORMAL ARCHITECTURES

A Methodology to Address SoS Challenges

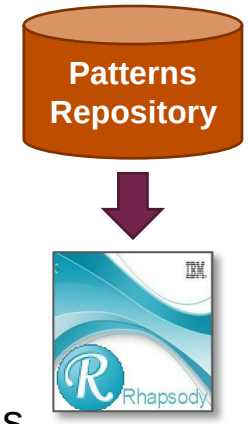
SoS Challenges

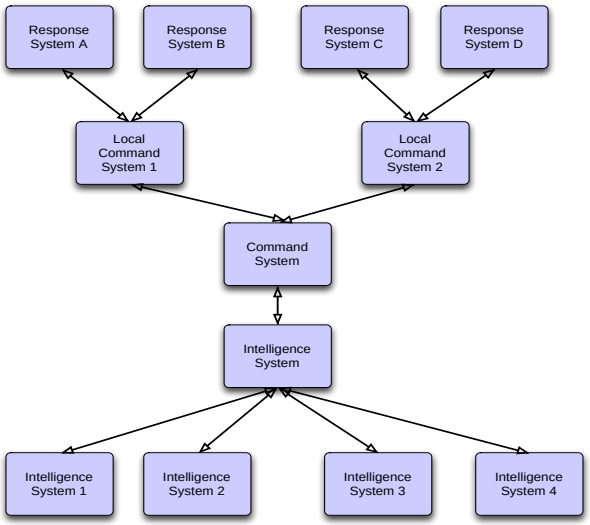
- Operational independency
- Heterogeneous Systems Integration
- Constituent Systems cooperation
- Design process complexity
- Reliability and Safety
- **Robustness to dynamicity**
- **Capability of evolution**
- Identification of Emergent Behaviors and capability to react to them
- …

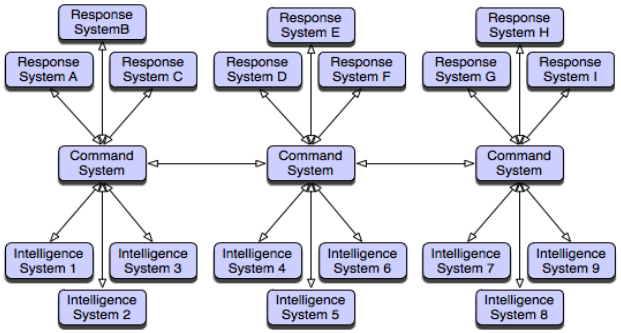


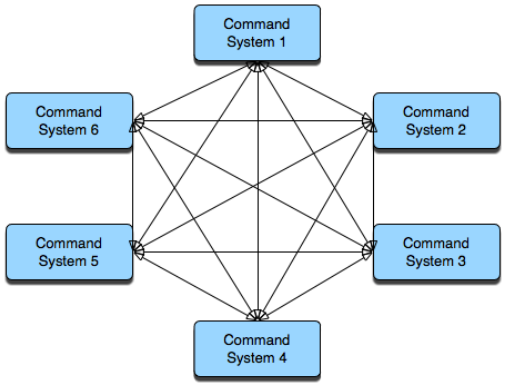
ARCHITECTURAL PATTERNS

- Design patterns used extensively in SW Eng.
- Architecture Patterns
  - A new approach to **reduce** Architecture-design **start-up time**
  - Capture **practitioner experience**
  - Are **template solutions** to known problems
  - Can provide **guidelines or implementation rules**
  - Allow to **compare** advantages coming from **different architectural approaches**
  - Are presented in an abstract way to **facilitate understanding** of complex architectures



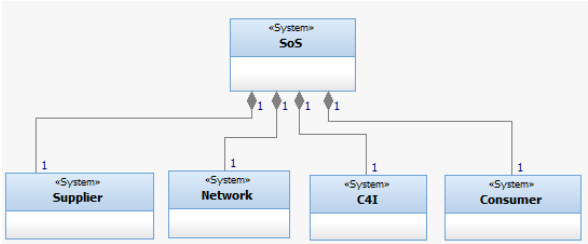




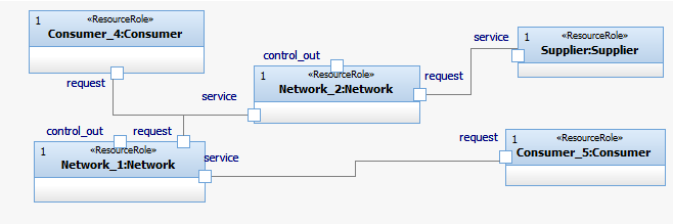


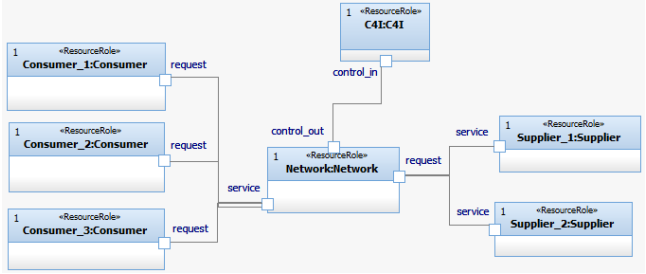
ARCHITECTURAL MODELLING (UPDM 2.0)

- ✓ **Hierarchy Relations** (BDD Diagram for System blocks **types**)



- ✓ **Architecture Instantiation** (IBD Diagram for System blocks **instances** and Net-List)

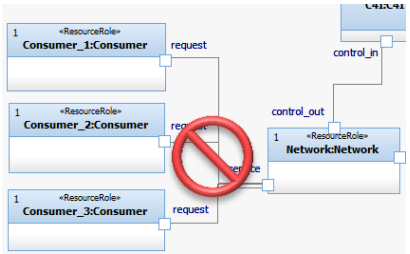




- ✓ **Architecture Dynamicity** (Graph Rewriting **Rules** for Reconfiguration)

- ✓ **Constrained** (e.g. by using OCL or OPL)

$$Network.connected(Consumer) \rightarrow sum() \leq 2$$

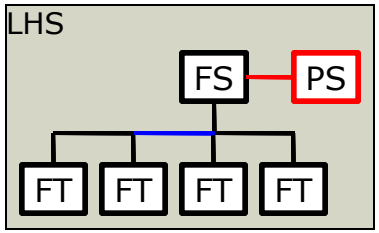


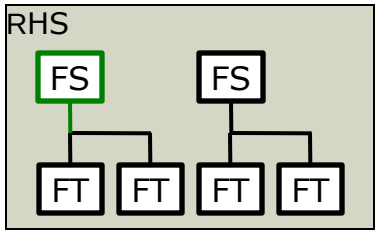
- ✓ **Extensible** (by defining **domain-specific** modeling **concepts**)

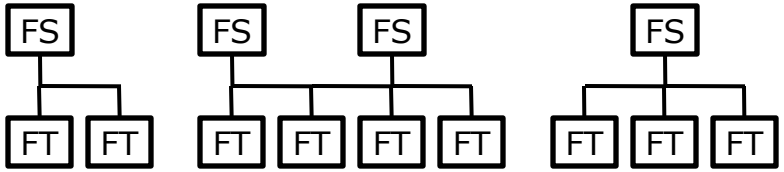
ARCHITECTURE VARIANTS GENERATION

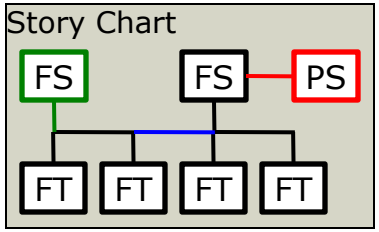
- Rules for changing the form of a set of relationships
  - **Left hand side** (LHS) depicts a pattern that can be matched
  - **Right hand side** (RHS) depicts a transformed version
  - **Story Chart** combines LHS and RHS into a transformation rule
- Any successful find of the LHS pattern can be replaced with the RHS
- *This method can automatically generate new architectures*

1. **Reader**: Matched, not changed.
2. **Eraser**: Matched and **removed**.
3. **Creator**: **Added** to the model.
4. **Embargo**: **Prevents the match**.





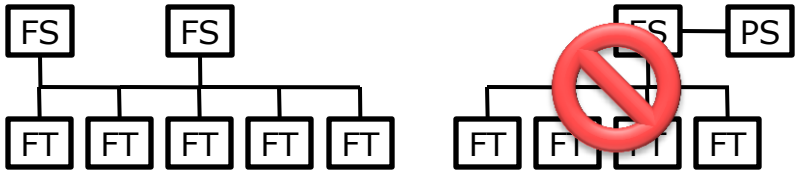




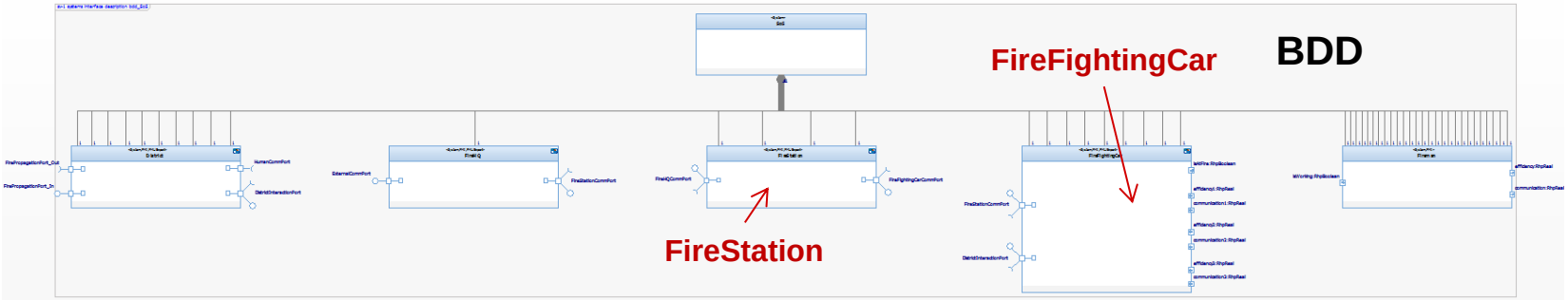
FT: Fire truck

FS: Fire station

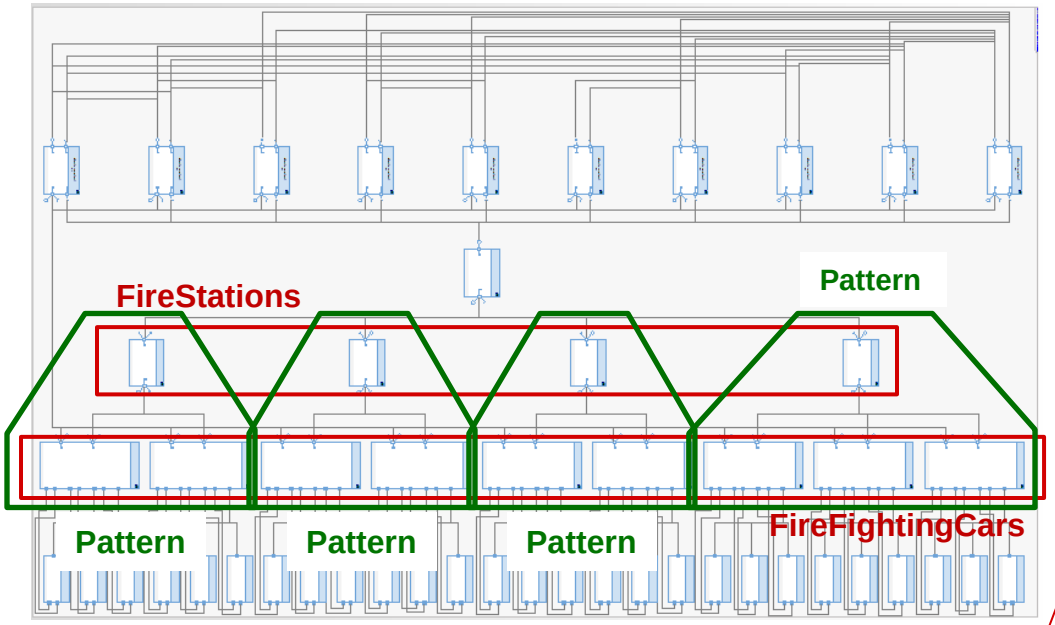
PS: Police station



COMBINING TEMPORAL AND ARCHITECTURAL PATTERNS



From the BDD diagram by instantiating systems we can obtain very complex IBD diagram:



To avoid error-prone and expensive repetition of the Requirements, we adopt OCL-based architectural patterns specification: a very expressive tool, if combined with temporal patters.

```
SoS.its(FireStation)->forAll(fireStation |
    whenever [
        fireStation.connected(FireFightingCar)->forAll(fs |
            fs.isAtFireStation = false)
    ] occurs [
        fireStation.connected(FireFightingCar)->exists(fs |
            fs.isAtFireStation = true)
    ] occurs within [0, inf)
    )
```

Temporal GCSL pattern

GCSL BEHAVIORAL – OCL

```
OCL-prop ::= true | false | path | not OCL-prop | OCL-prop b-relop OCL-prop |
        OCL-expr a-relop OCL-expr |
        if OCL-prop then OCL-prop else OCL-prop |
        OCL-coll -> forAll( variableName | OCL-prop ) |
        OCL-coll -> exists( variableName | OCL-prop ) |
        OCL-coll -> empty() |
        OCL-coll -> notempty()

b-relop ::= and | or | xor | implies | iff
a-relop ::= > | >= | = | <> | =< | <

OCL-expr ::= ElapsedTime |
        path | OCL-coll -> sum() | OCL-coll -> size() |
        if OCL-prop then OCL-expr else OCL-expr |
        const | - OCL-expr | ( OCL-expr ) |
        OCL-expr a-binfn OCL-expr

a-binfn ::= + | - | * | /

OCL-coll ::= itsClassNames | its(ClassName) |
        connectedClassNames | connected(ClassName) |
        basicTypeAttributeName |
        objectName . OCL-coll | variableName . OCL-coll |
        itsClassNames . OCL-coll | its(ClassName) . OCL-coll

path ::= basicTypeAttributeName | objectName . path | variableName . Path
```

Quantification on OCL-defined sets (collections) of sub-portions of the architecture

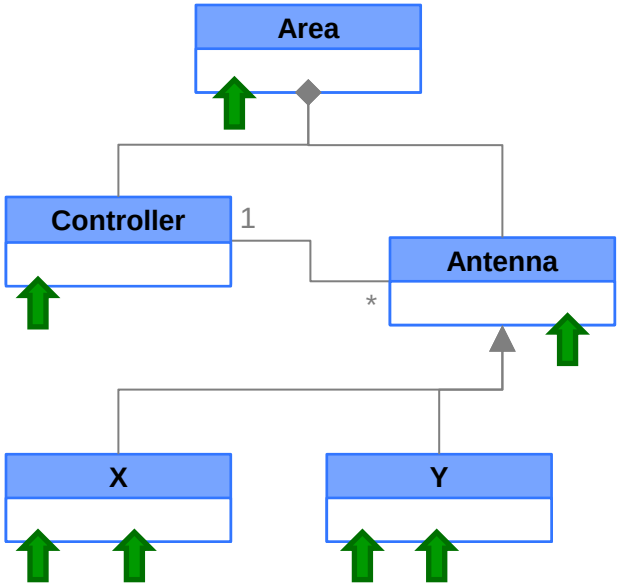
+

Expressive constraints on System blocks attributes

+

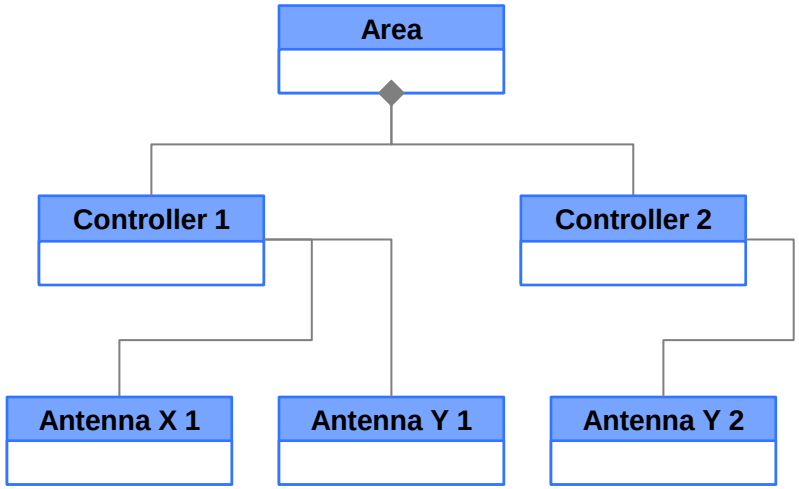
OCL collections defined through SoS-specific hierarchy/connection navigation predicates

ARCHITECTURE OPTIMIZATION (DANSE EXAMPLE)



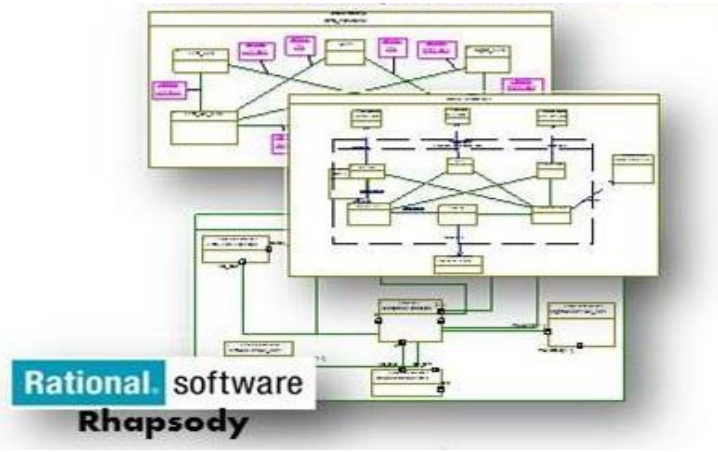
1. Geographic areas contain controllers and antennas
2. Each controller has a position and controls many antennas (within a range)
3. An antenna has a position
4. Antennas come in two types
5. X antennas have a higher bandwidth lower a lower coverage
6. Y antennas have lower bandwidth and higher coverage

- Green arrows indicate **numerical parameters** (sqkm, cost, coverage, bandwidth …)
- Purpose is to **maximize Area coverage** and network **bandwidth** while **minimizing cost**
- **Number and type of antennas** is variable
- There are several possible **optimal architectures**



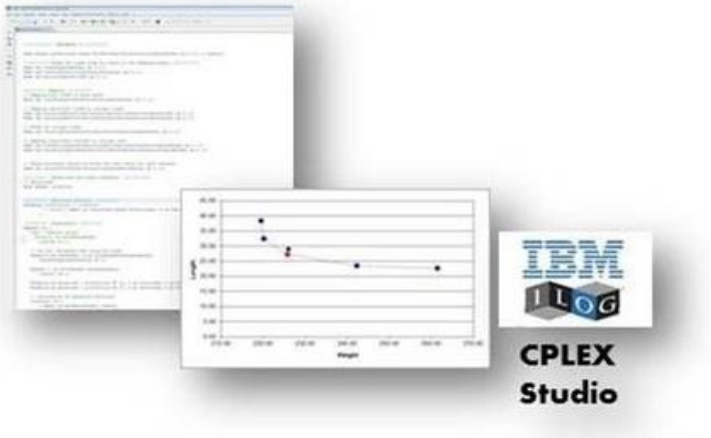
DANSE Approach to Architecture Optimization

1. Describe SoS pattern through different SysML views

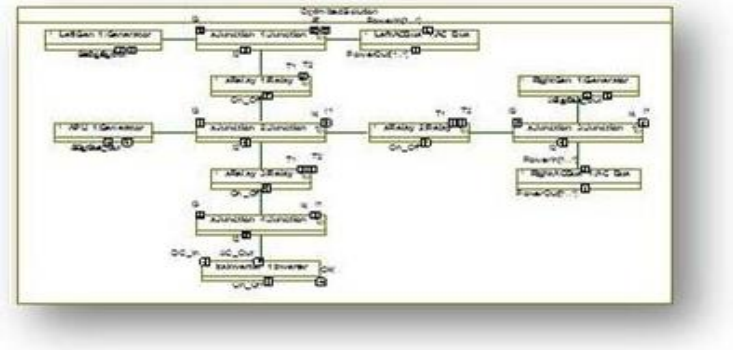


2.Provide Data Input and Output structure

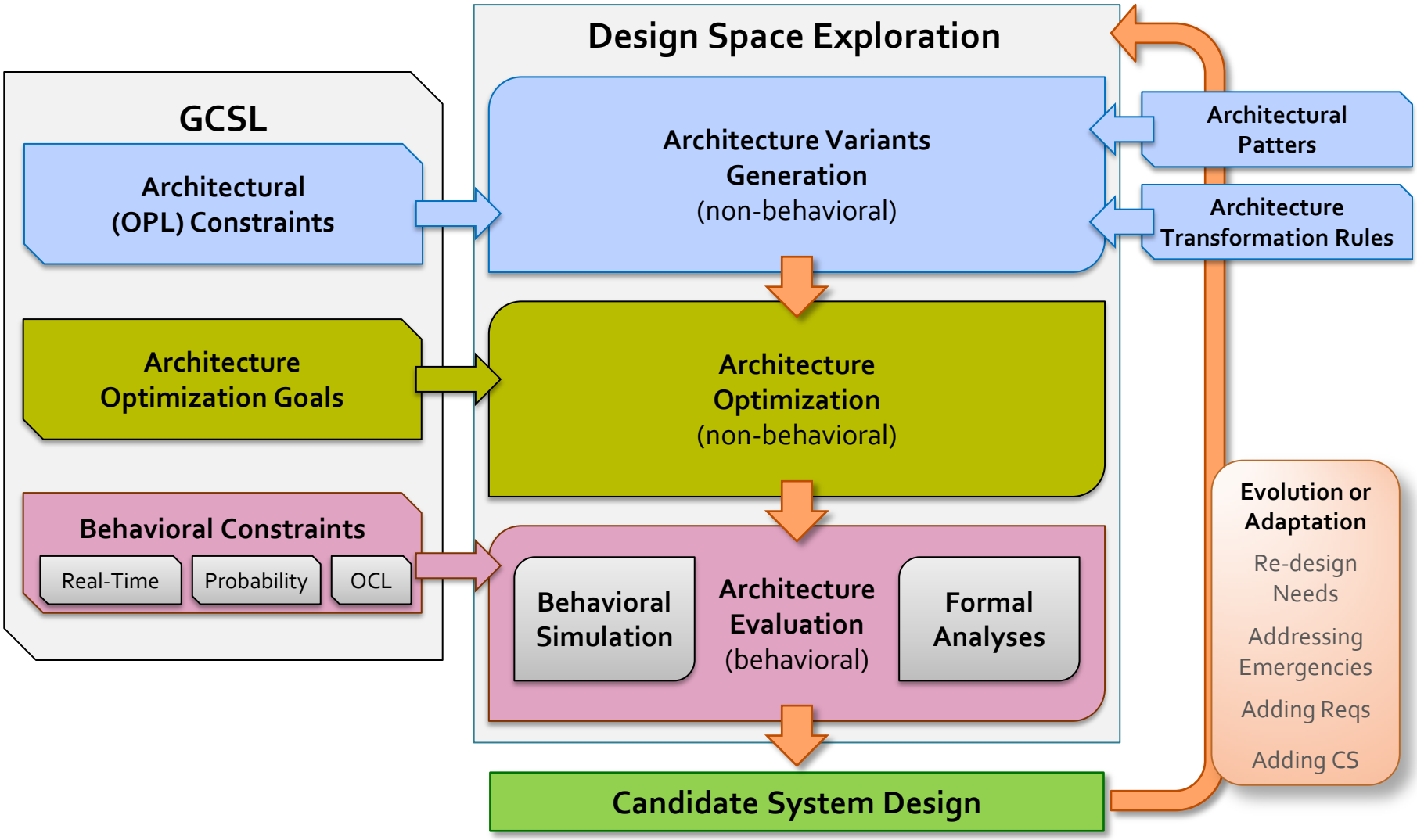
3. Automatic translation into optimization solver



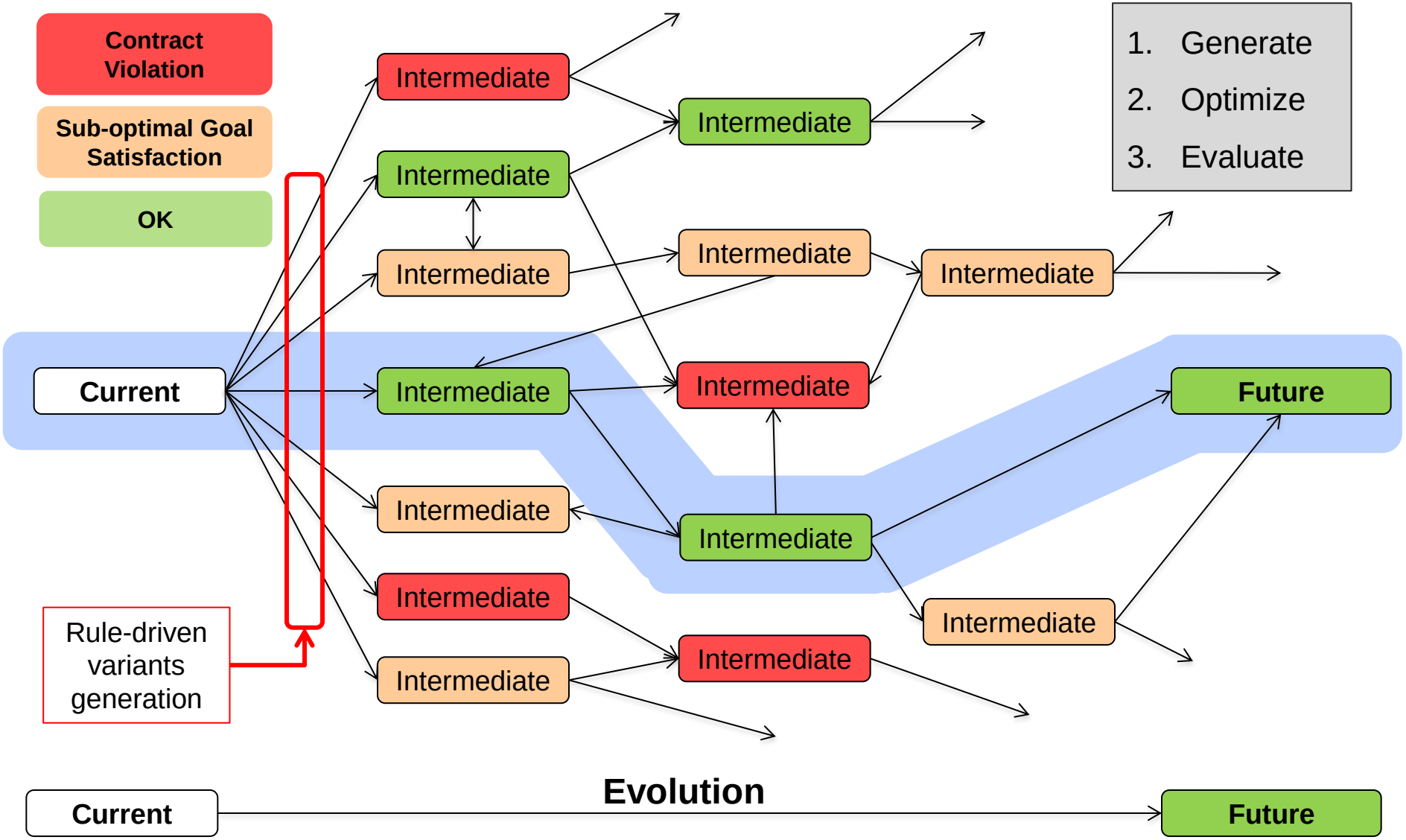
4. Optimized system back annotated to SysML model



Analyses Support to Evolution



Design Space Exploration for Evolution



FORMAL ARCHITECTURES

Challenges:

1. Complexity coming from several and layered Views covering different aspects of the architecture
2. Need to evaluate cost and efficiency beforehand
3. Managing Variants coming from :
   evolution, product families, alternative solutions or system dynamicity

Advantages of formal architectures:

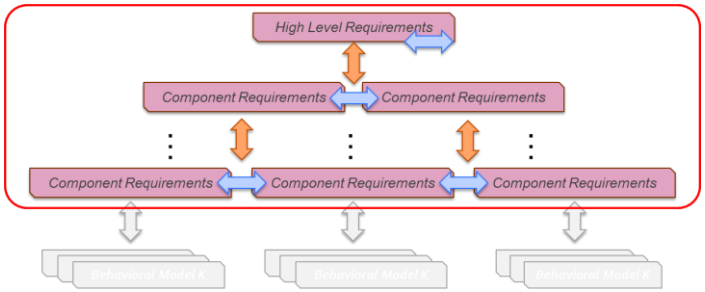
1. **Pattern libraries** capturing domain expertise can be seamlessly integrated to enable reuse
2. Modeling **Guidelines** can be formalized and enforced as constraints
3. Design generic **Instantiable** architectures (with parametric number of components)
4. Formal constraints and costs allow automated **Architecture Optimization**
5. Formal architecture reconfiguration rules allow to model **System Dynamicity** and **Product Families**
6. Support **systematic** Architecture Variants **exploration and evaluation** to realize evolutionary and adaptive behaviors

FORMAL ANALYSES

ANALYSES POSITIONING

1. Requirements Analyses

- Assumptions: Architecture defined and Requirements Allocated
- **Consistency**: Assessing if requirements can be implemented
- **Dominance**: Assessing correct decomposition
- Positioning: Applicable at every decomposition step

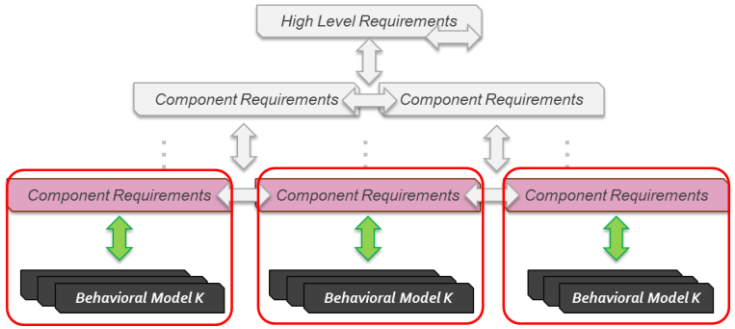


2. Behavioral Simulation

- Assumptions: Decomposition completed (may be further refined) and Behavioral Model available
- Role: Supporting better understanding of system behavior and a first measure of efficiency
- Positioning: Applicable at the end of each design cycle

3. Satisfaction Analysis

- Assumptions: Same as Behavioral Simulation
- Role: Check correctness of Behavioral Model against Formal Specifications
- Positioning: Same as Behavioral Simulation



4. Resilience Analysis

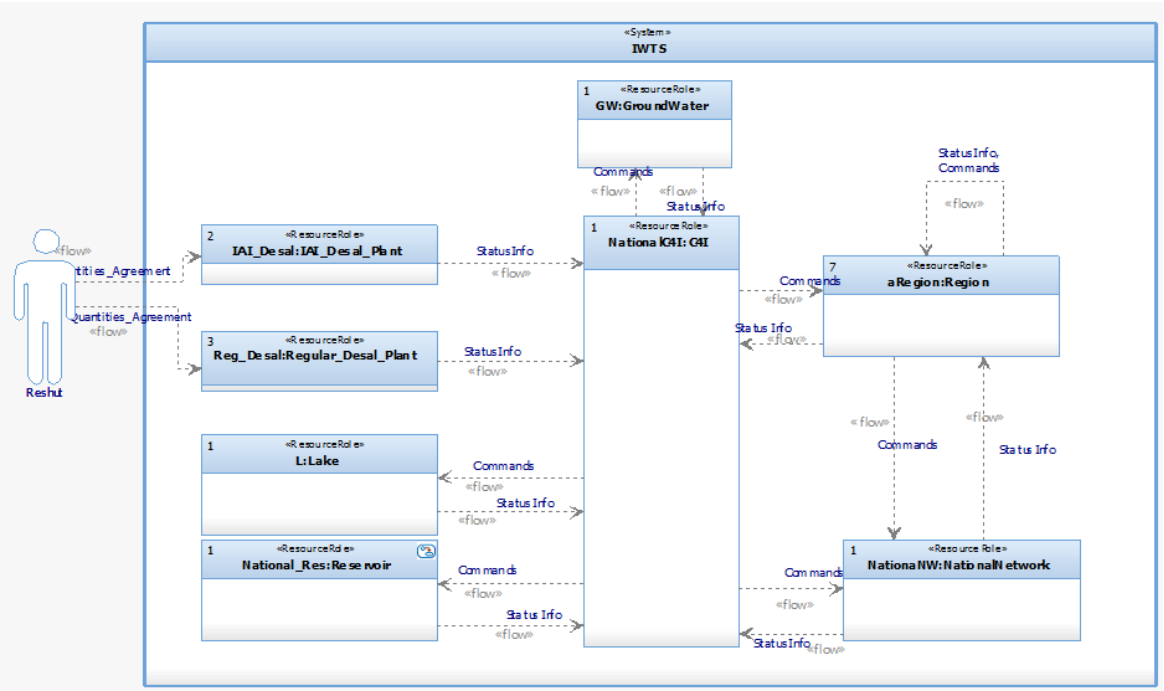
- Assumptions: Abstract Behavioral Model available, including Failure Modes
- Role: Assess effectiveness of counter-measures in case of faults
- Positioning: Early design cycles completion

DANSE - WATER TREATMENT AND SUPPLY

Ground Water

Water Production

Water Reserve



Water Consumption

Water Transport

IWTS Control

IWTS Constraints:
ensure water **availability** and **quality** across the year, leveraging water reserves and seasonal flows

IWTS Challenges:
1. ensuring **reliability** and **availability**
2. ability to evolve to follow change of **water needs** and **season behavior**,

IWTS – Requirements Analysis – DANSE Case Study

IWTS: **A nation-wide system for water treatment and supply**

A pipe burst may constitute a potential danger for the effective supply of water

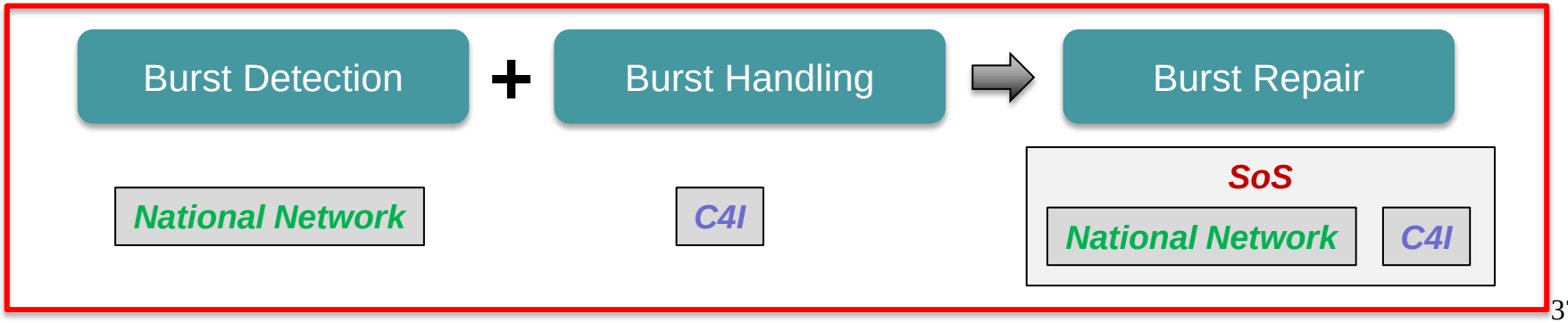
Scenario: Pipe burst handling and repair

Analysis: *Do burst handling and repair time ensure limited disruption time?*

Constituent Systems Involved:

- National Network
- C4I centralized control

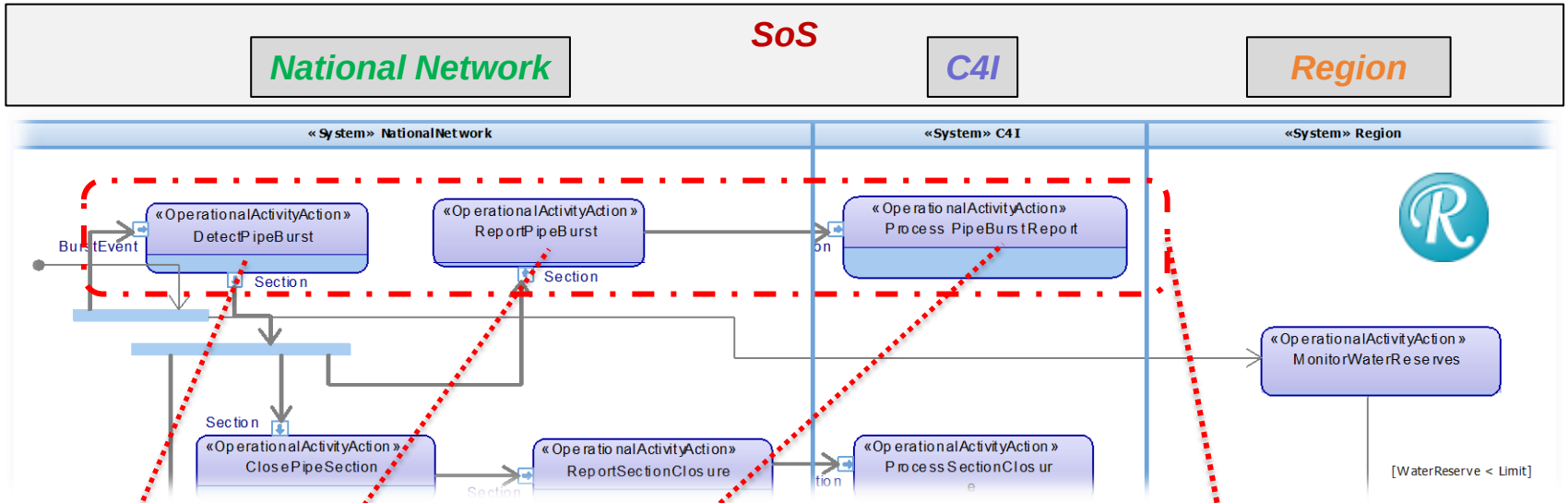




IWTS – Requirements Analysis - Decomposition

Identify activities of interest in the Activity Diagram (OV-5b UPDM)

1



ReportPipeBurst *activity should last between* ***5 and 10 minutes***

From ***Burst*** *to* ***ProcessReport*** *the activity should last between* ***20 and 45 minutes***

DetectPipeBurst *activity should last between* ***1 and 4 minutes***

ProcessPipeBurstReport *activity should last between* ***15 and 30 minutes***

Allocate requirements to SoS

3

Allocate requirements to single CSs

2

IWTS – REQUIREMENTS ANALYSIS - RESULTS

Relax
Global Req

SoS	**whenever [** NationalNetwork.eventPipeBurst **] occurs [** C4I.eventReportToUser **] occurs within [** 20 min , 45 min **]**
C4I	**whenever [** C4I.eventBurstReport **] occurs [** C4I.eventBurstHandling **] occurs within [** 15 min , 30 min **]** **whenever [** C4I.eventBurstHandling **] occurs [** C4I.eventReportToUser **] occurs within [** 1 min , 2 min **]**
National Network	**whenever [** NationalNetwork.eventPipeBurst **] occurs [** NationalNetwork.eventBurstDetection **] occurs within [** 1 min , 4 min **]** **whenever [** NationalNetwork.eventBurstDetection **] occurs [** NationalNetwork.eventBurstReport **] occurs within [** 5 min , 10 min **]**

46 min

3 min

Strengthen
Local Req

Counter-example :

	Time (s)	eventPipeBurst	eventBurstDetect	eventBurstReport	eventBurstHandling	eventReportToUser
4 min	0.0	T	F	F	F	F
13 min 59 sec	240.0	F	T	F	F	F
43 min 1 sec	839.0	F	F	T	F	F
45 min 1 sec	2581.0	F	F	F	T	F
	2701.0	F	F	F	F	T

Counterexample analysis an

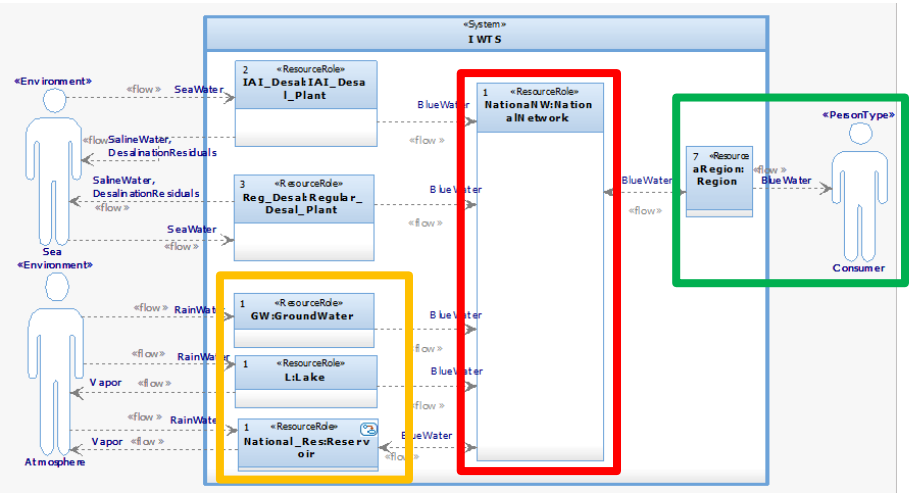
Value provided by the Analysis:

1. Validation of SoS-level performance requirement decomposition into Constituent Systems requirements
2. Design of C4I and NationalNetwork architectures and behavioral models can now be performed independently, maintaining

IWTS – Behavioral Simulation – DANSE Case Study

The Command and Control Center is responsible for **orchestrating water delivery** to regions.

Different strategies for **activating & deactivating Desalination plants** may influence the overall system **performance**.

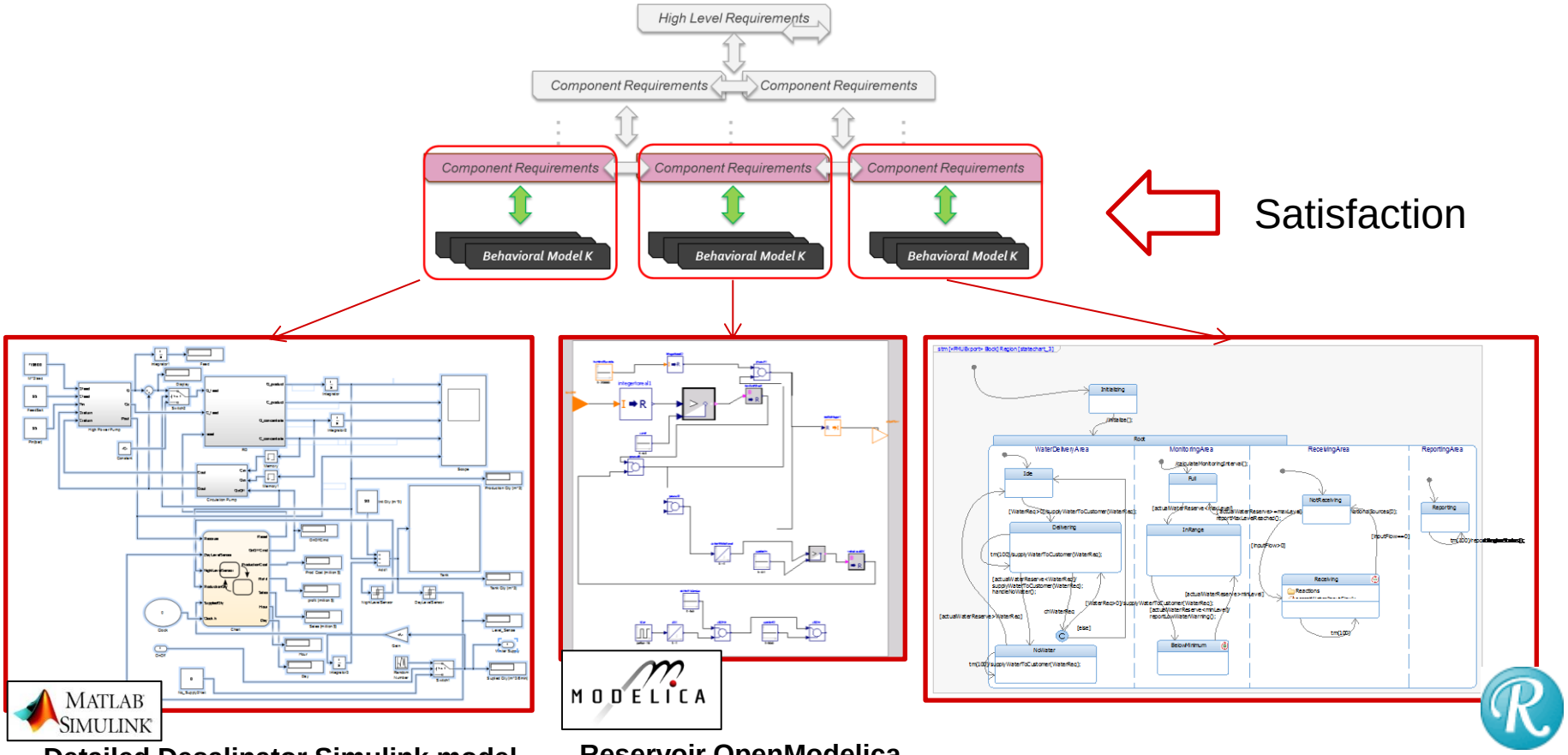


Water needs vary with **increasing population size**. The IWTS system should adapt and produce a sufficient amount of water to meet the demand while meeting budget constraints.

Production plans **tuning** may lead to higher efficiency and reduced costs.

Detailed behavioral evaluation of the System can be performed as soon as behavioral models are available. Integration of behavioral authoring tools is a challenge that we address using simulation based on the FMI standard.

HETEROGENEOUS BEHAVIORAL MODELS



Detailed Desalinator Simulink model

Reservoir OpenModelica Behavioral Model

Rhapsody State-chart for region

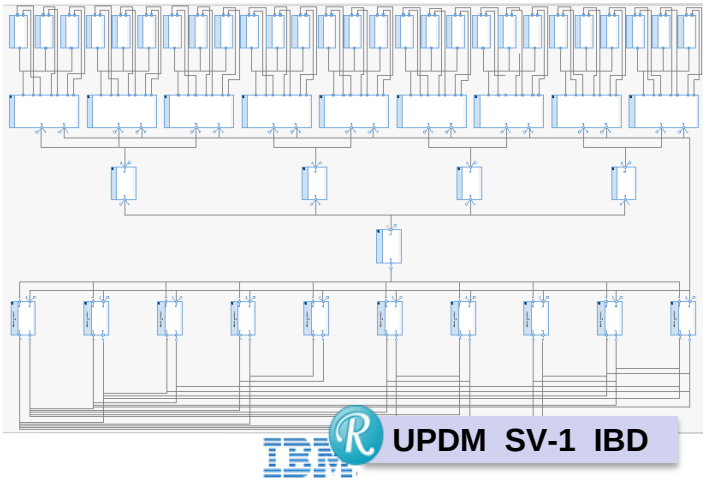
Challenge :

Constituent Systems' behavioral models designed using distinct authoring tools.

BEHAVIORAL MODELS INTEGRATION FLOW

Sources:

- File System
- Semantic Mediation Container



FMI standard

Behavioral Models

MATLAB SIMULINK | MODELICA | R

FMU Export

FMUs

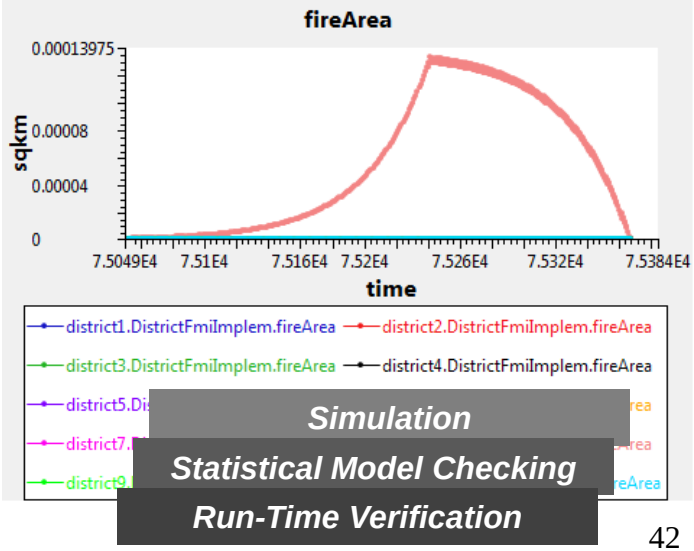
SoS Architecture & Configuration

DANSE UPDM Export

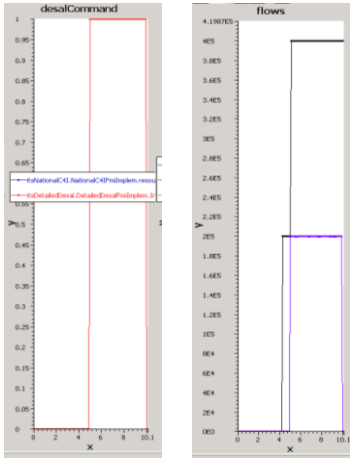
Intermediate Format

DESYRE IMPORT

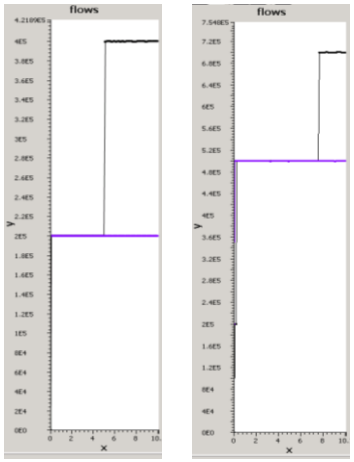
Integrated Model for Analysis



SIMULATION SUPPORTING DESIGN-TIME DECISIONS

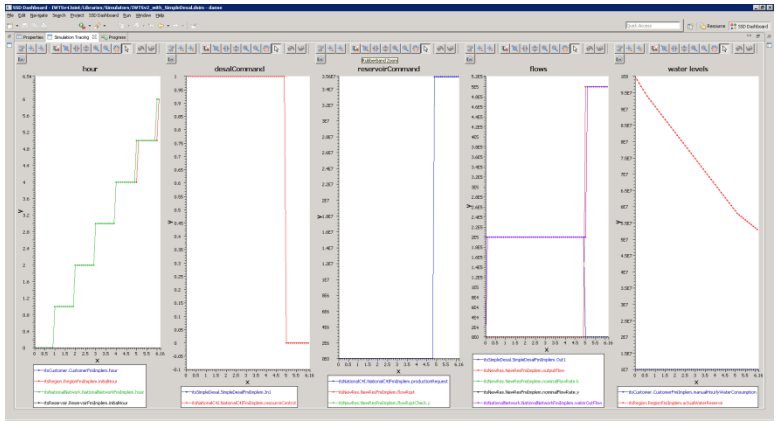


50 % DUTY-CYCLE

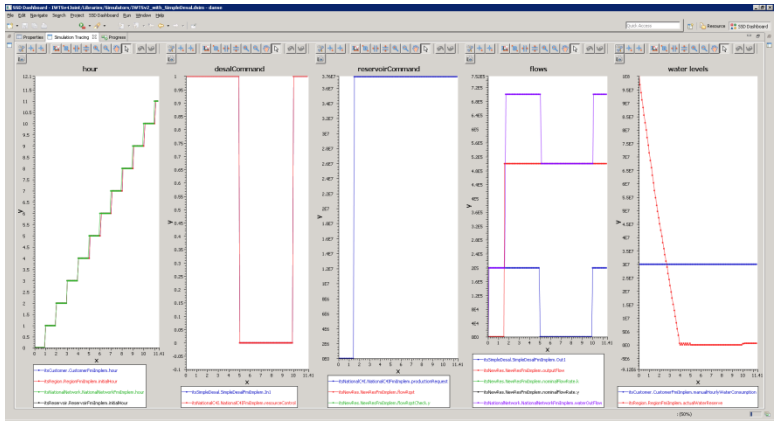


100 % DUTY-CYCLE
RE-TUNING

Different **C4I control strategies** have impact on Flows efficiency



IWTS behavior with **current water demand**



IWTS behavior with **future water demand**

REQUIREMENTS SATISFACTION ANALYSES

1. **Contract-based Run-Time Verification (RTV)**
   - GCSL Contract translated into an FMU component
   - New FMUs integrated into simulation model
   - Compose monitor traces to test Satisfaction & Dominance
   - Use FMU monitors on Deployed System for Run-time Verification

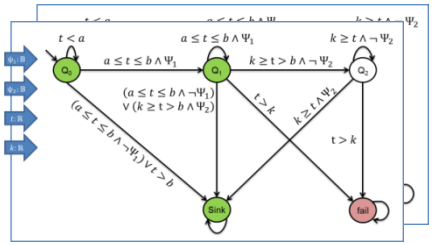
2. **Statistical Model Checking (SMC)**
   - Run thousands of simulation (Simulator driven by SMC tool)
   - Monitor traces, providing satisfaction/violation flags
   - Aggregate satisfaction/violation data and perform statistical analysis
   - Collect results into an estimate of the probability of Contract Satisfaction, including indication on Precision and Confidence

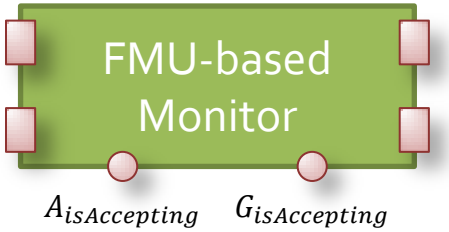
FROM GCSL TO MONITORS

GCSL Assumption and Guarantee assertions

$(\Psi_1 \text{ during } [a, b] \text{ raises } \Psi_2)$

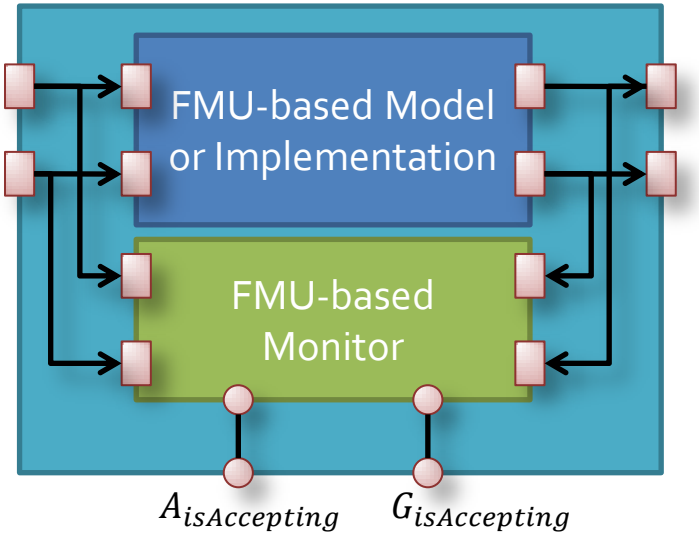
Assertions State Machines



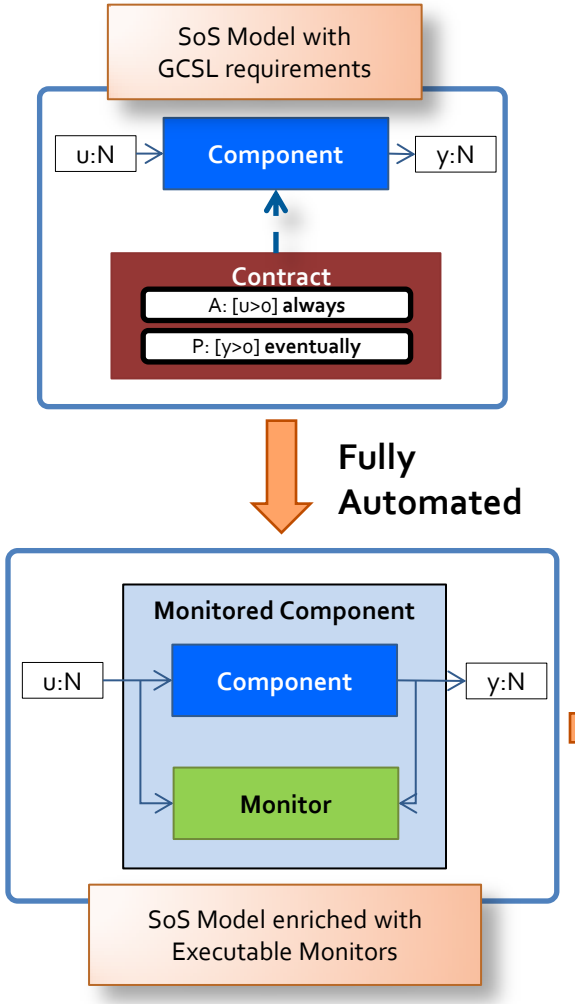


$$A_{isAccepting} \Leftrightarrow trace \in Assumption$$
$$G_{isAccepting} \Leftrightarrow trace \in Guarantee$$

- GCSL state-machine based semantics is used to define the monitor behavior
- State machines for Assume & Guarantee are composed and compiled into an FMU
- FMU outputs denote A/G satisfaction and can be composed to check various contract relations
- The monitor is attached to the component

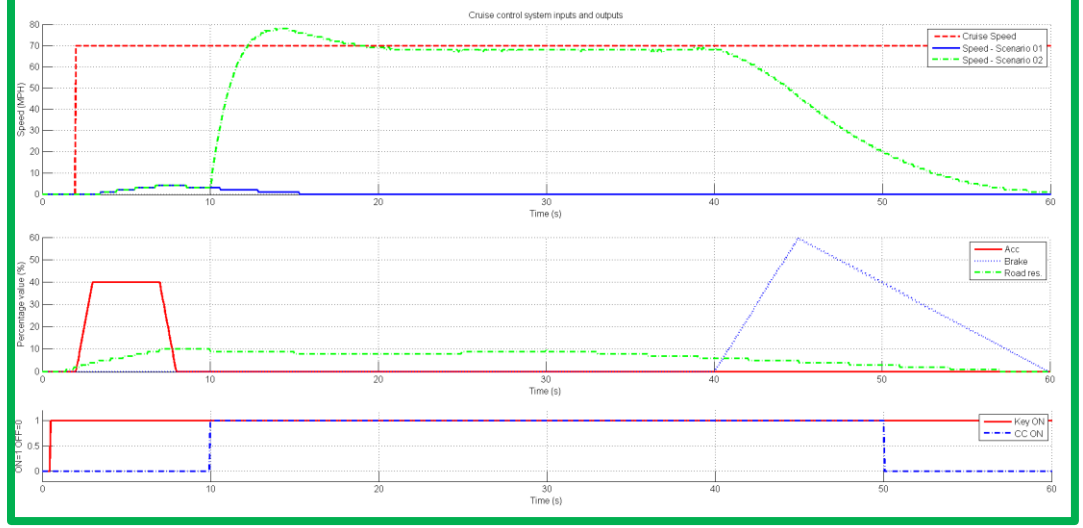


CONTRACTS-BASED RUN-TIME VERIFICATION

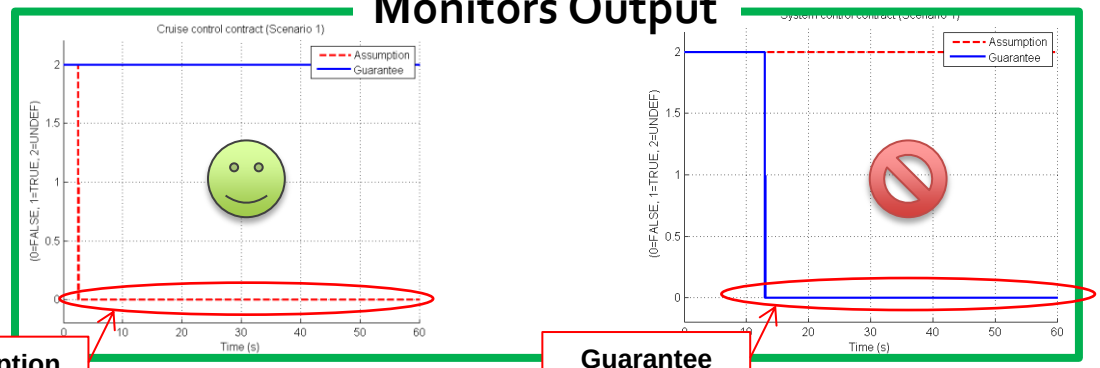


GCSL requirements are embedded into the SoS Architecture (synthesis of monitors) and observed in simulation

Model Simulation



Monitors Output



Assumption

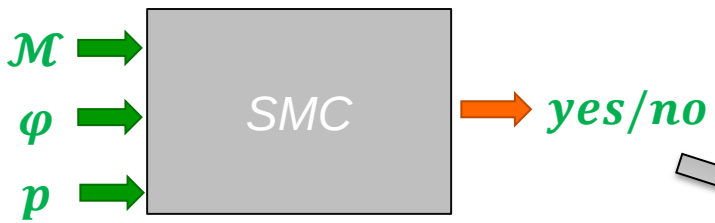
Guarantee

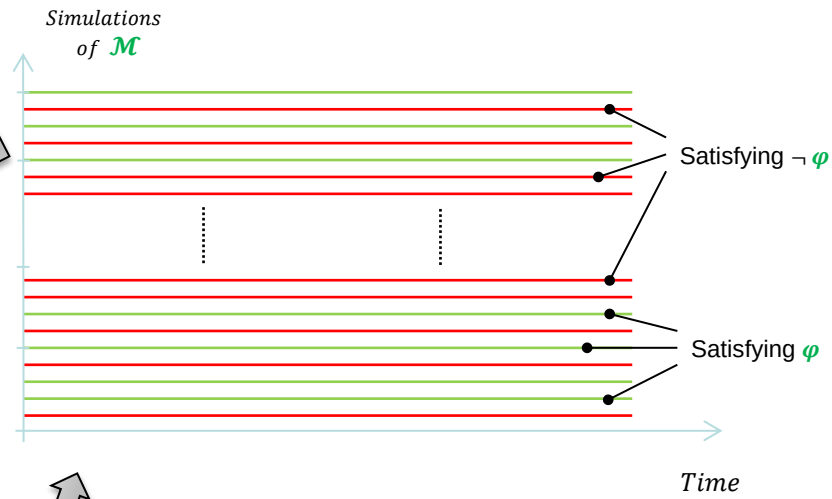
SMC – A TUNABLE FORMAL ANALYSIS

Qualitative question

$$\mathbb{P}(\mathcal{M} \models \varphi) >_? p$$

Evaluation of a property against a given probability threshold and with a minimal number of simulations





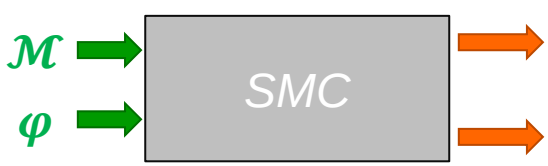
Quantitative question

$$\mathbb{P}(\mathcal{M} \models \varphi) = ?$$

Analysis results accuracy

(precision) ϵ

(confidence) $1-\delta$



Minimal number of simulations

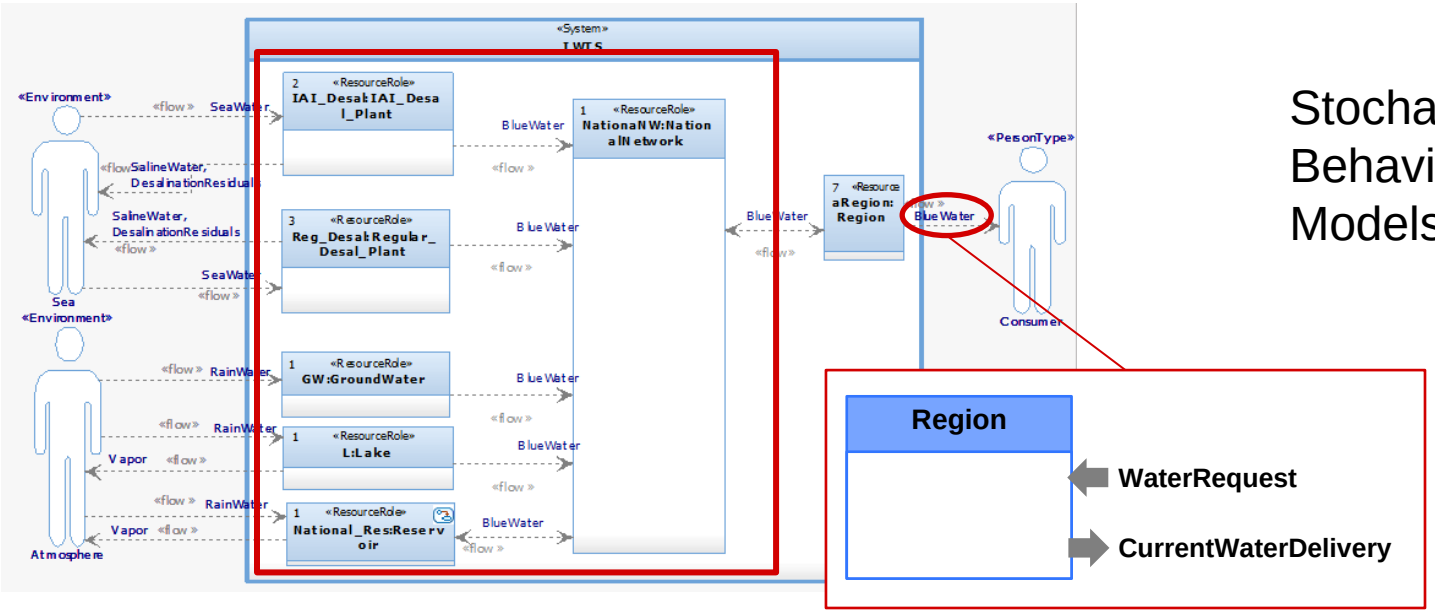
Analysis cost

$\hat{p}$

$$\mathbb{P}(|\hat{p} - p| \leq \epsilon) \geq 1-\delta$$

Evaluation of a property with no given probability but with fixed analysis cost

IWTS – Requirements on Customer Satisfaction



Stochastic FMU Behavioral Models

- The Region provides water according to Customers' needs
- The region water availability depends on the water delivered by the IWTS NationalNetwork
- NationalNetwork production depends from the behavior of several Constituent Systems and a myriad of season-dependent and history-dependent factors

Requirement: guarantee sufficient water delivery to customers

always [Region.CurrentWaterDelivery >= region.WaterRequest]

at the end of [0,24], [region.currentWaterDeliveryQuantity >= region.WaterReq] has been true at least [90] % of time

IWTS – Statistical Model Checking

Requirement 1

always [Region.CurrentWaterDelivery >= region.WaterRequest]

Strong requirement: each simulation point in time should satisfy it.

Does not allow minimal deviation for exceptional events (e.g. pipe burst).

Since we granted limited disruption time (through Requirements analysis) we might weaken it as follows.

Requirement 2

at the end of [0,24], [region.currentWaterDeliveryQuantity >= region.WaterReq] has been true at least [90] % of time

More permissive version: the requirement is satisfied a sufficient percentage (90%) of the total time (***24 hours***). Few minutes water shortage is considered as acceptable.

Analysis Settings:

1. **Probability of incorrect result is set to 0.01**, thus obtaining a confidence of 99%
2. **Precision = 0.1**

$$\mathbb{P}(\mathcal{M} \vDash Requirement_1) \in [0.76, 0.96]$$
$$\mathbb{P}(\mathcal{M} \vDash Requirement_2) \in [0.8, 0.1]$$

3. **Precision = 0.05**

$$\mathbb{P}(\mathcal{M} \vDash Requirement_1) \in [0.83, 0.93]$$
$$\mathbb{P}(\mathcal{M} \vDash Requirement_2) \in [0.84, 0.94]$$

Results Summary

Precision	0.1	0.05
Confidence	99%	99%
Number of simulations	265	1060
Probability of Contract 1	0.86	0.88
Probability of Contract 2	0.90	0.89
Analysis time	6 min	25 min

Quick evaluation may lead to the decision of activating a new Desalinator

INTEGRATION SUPPORT

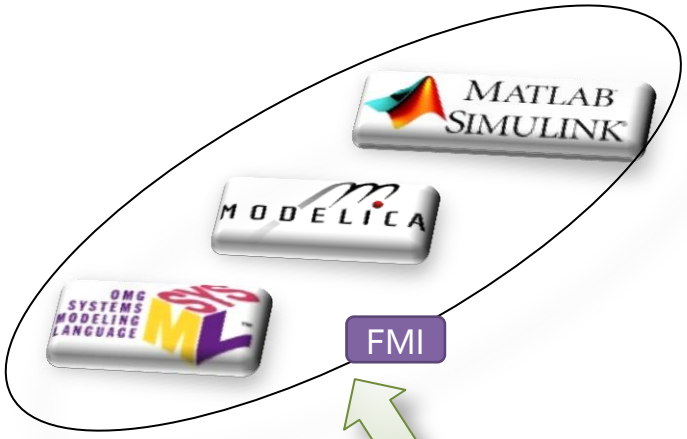
A Methodology to Address SoS Challenges

SoS Challenges

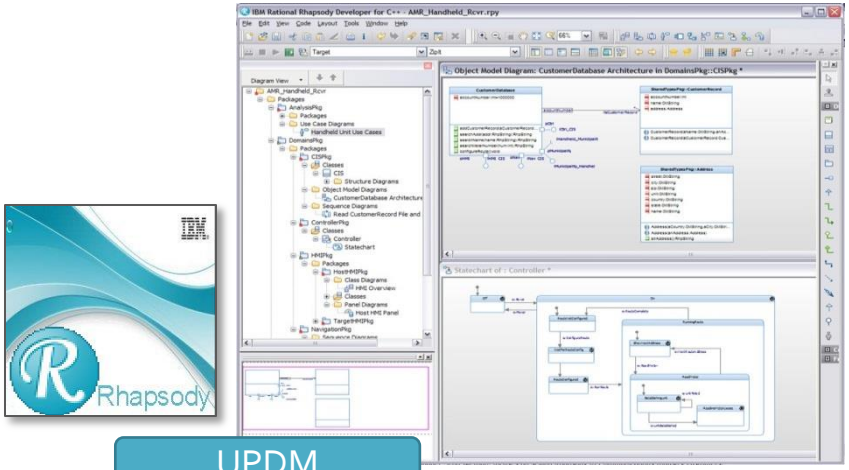
- Operational independency
- Heterogeneous Systems Integration
- Constituent Systems cooperation
- Design process complexity
- Reliability and Safety
- Robustness to dynamicity
- Capability of evolution
- Identification of Emergent Behaviors and capability to react to them
- …

THE DANSE TOOL-NET

Implements Semantic Mediation



FMI



UPDM

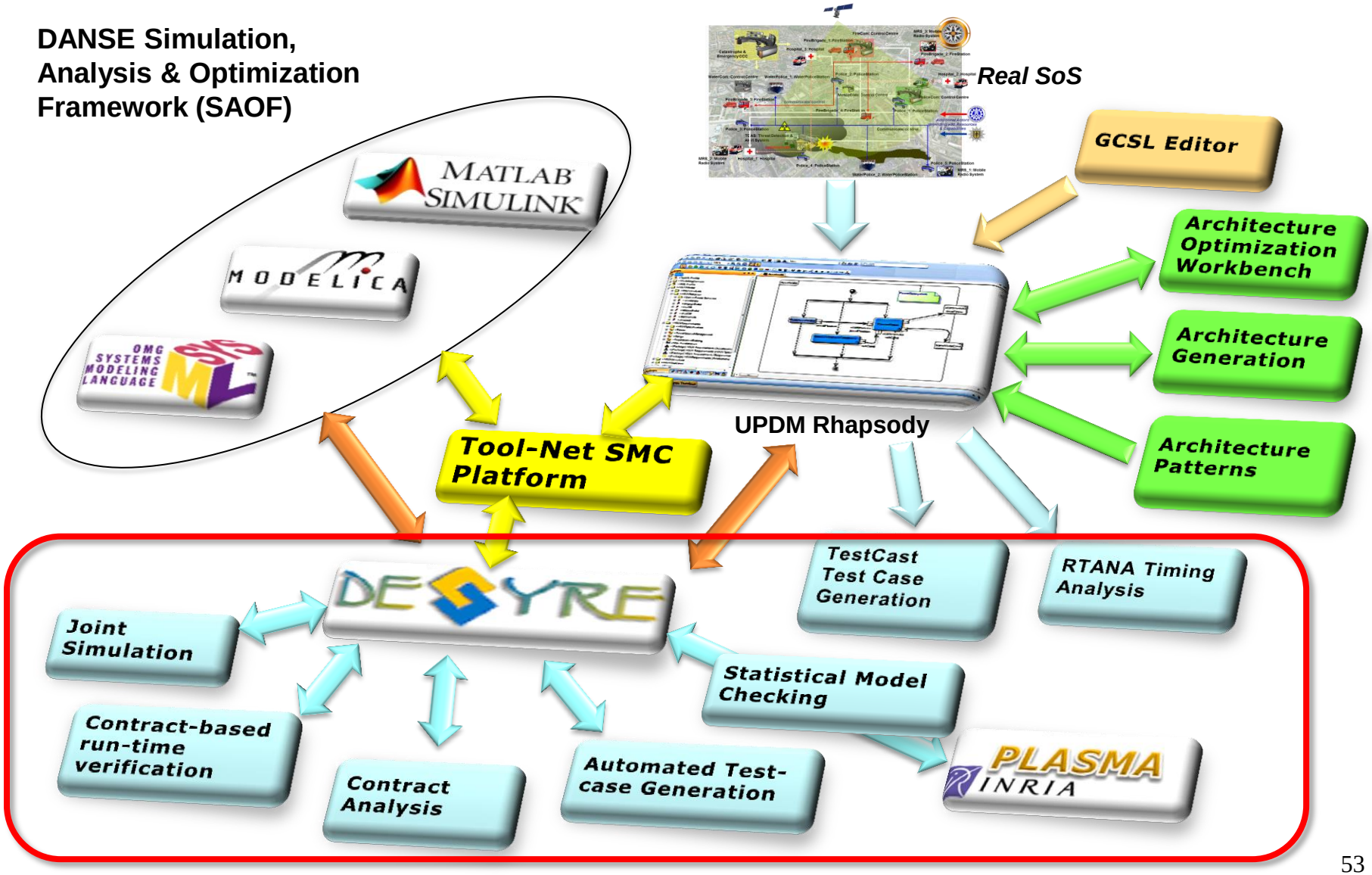
DANSE extension

DANSE Semantic Mediation -based Tool-Net

OSLC

DANSE Analysis Tools

DANSE Integrated Framework



THANK YOU