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
UTC BUSINESSES (2014)

Commercial

OTIS

2014 net sales $65.1 billion

Aerospace

Sikorsky

UTC Aerospace Systems

UTC Systems & Controls Engineering

ALES S.r.l. - This page contains no technical data subject to the EAR or the ITAR - Approved for public release
DANSE EU PROJECT FACTS

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

![Consortium Logos]

- **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)
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

1. Design complexity mastered through decomposition
2. Architecture evolution models capture evolution capabilities
3. Requirements and Architecture improved using feedback from lower design levels
Formal Analysis provides valuable feedback during design.

When decomposition is complete

Requirements feedback

Architecture feedbacks

- Requirements feedback
- Architecture feedbacks

Formal Analysis

Layer i

(Sub-)System Requirements

Architecture Definition

Architectures

Formal Analysis

Layer i

Component Specification

Component Specification
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)

• Formal Analysis role:
  **local** assessment against (formal) Specifications entails composite system correctness against **global** Requirements

### Diagram:

- **Component Specification**
- **Formal Analysis**
- **Behavioral Model 1**
- **Behavioral Model 2**
- **Behavioral Model K**
- **Robust Design**
- **Speculative Analysis**
DANSE MBSE METHODOLOGY FOR SoS

- Modelling
  - Change Implementation
  - Observation
  - Operation
  - SoS Initiation Phase
  - SoS Creation Phase
  - SoS Operation Phase (continuous)
  - Model SoS behaviour
  - Operate the SoS
  - Define potential needs
  - Analyze possible architecture changes
  - Influence and implement changes

- SoS Engineering
- Capability Learning Cycle
- Constituent Systems Engineering
- DANSE Technologies
  - SoS language
  - Architecting
  - SoS itself
  - Simulations, predictions
  - SoS observations
  - Simulations
  - Analysis, optimization
  - Forecast behaviors
  - Coordinated monitoring
  - Traditional systems engineering to create/modify systems
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 $\Rightarrow$ Guarantee

(if applicable, it ensures…)

- Contracts are more than types: they are behavioral descriptions

```
<table>
<thead>
<tr>
<th>IN</th>
<th>1.3</th>
<th>0.2</th>
<th>1.0</th>
<th>7.3</th>
<th>7.8</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
</tbody>
</table>
```

Rejected Trace

```
<table>
<thead>
<tr>
<th>IN</th>
<th>3.1</th>
<th>2.0</th>
<th>1.5</th>
<th>4.3</th>
<th>6.1</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>...</td>
</tr>
</tbody>
</table>
```

Accepted Trace
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 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

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

Everytime [reqTowBot] then [reqServed] follows

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

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

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
GCSL: A SoS-specific Formal Requirements 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

**Contracts specification language**

Requirements on
- # of components,
- # of connections,
- weight, cost, length, ...

Objective functions on architecture cost parameters

SoS and CSs (abstract) behavior description

Architectural Constraints

Optimization Goals

GCSL provides a multi-view specification language
GCSL BEHAVIORAL - FEATURES

✓ Real-Time, Numeric Constraints, Probability
  within 3.1 hours, $\text{Component}_1.\text{port}_3 \geq \text{Component}_2.\text{port}_1 + 3$, with probability 0.99

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

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

✓ 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)

Natural-language based specification patterns

whenever \[ \text{fireArea} > 0 \] occurs
\[ \text{FireFightingCar.isAtFire} \] occurs within \[1 \text{ hour}\]

Usable formal requirements

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

With underlying formalized semantics
**Formal Analyses on Behavioural Requirements**

- **Horizontal partitioning & allocation**
  - Layer 1
    - CPS
  - Layer 2
    - Sub-System Requirements
    - Sub-System Requirements
  - Layer N
    - Component Requirements
    - Component Requirements
    - Component Requirements

- **Vertical refinement**
  - System Requirements
  - Sub-System Requirements
  - Component Requirements
  - Behavioral Model K
  - Behavioral Model K
  - Behavioral Model K

- **Consistency** = Existence of Implementation
- **Dominance** = Correct decomposition
- **Satisfaction** = Correct implementation

**Requirements Analyses**
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)
  \[ \text{Network.connected(Consumer)} \rightarrow \text{sum()} \leq 2 \]

- **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 patterns.

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
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
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**
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
DESIGN SPACE EXPLORATION FOR EVOLUTION

1. Generate
2. Optimize
3. Evaluate

Contract Violation
Sub-optimal Goal Satisfaction
OK

Current
Future

Rule-driven variants generation

Evolution

Generate
Optimize
Evaluate
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
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

---

ALES S.r.l. - This page contains no technical data subject to the EAR or the ITAR - Approved for public release
DANSE - WATER TREATMENT AND SUPPLY

IWTS Control

Ground Water

Water Production

Water Reserve

Water Consumption

Water Transport

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

1. **DetectPipeBurst** activity should last between 1 and 4 minutes
2. **ProcessPipeBurstReport** activity should last between 15 and 30 minutes
3. **ReportPipeBurst** activity should last between 5 and 10 minutes

Allocate requirements to single CSs

Allocate requirements to SoS

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

Counterexample:

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>eventPipeBurst</th>
<th>eventBurstDetect</th>
<th>eventBurstReport</th>
<th>eventBurstHandling</th>
<th>eventReportToUser</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>240.0</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>839.0</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>2581.0</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>2701.0</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
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

Challenge:
Constituent Systems' behavioral models designed using distinct authoring tools.
**Behavioral Models Integration Flow**

**Sources:**
- File System
- Semantic Mediation Container

**FMU Export**

**FMUs**

**DANSE UPDM Export**

**Intermediate Format**

**DESYRE IMPORT**

**Integrated Model for Analysis**

**FMI standard**

- Behavioral Models
- SoS Architecture & Configuration

**Simulation**

**Statistical Model Checking**

**Run-Time Verification**
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 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

\[ A_{\text{isAccepting}} \Leftrightarrow \text{trace } \in \text{Assumption} \]
\[ G_{\text{isAccepting}} \Leftrightarrow \text{trace } \in \text{Guarantee} \]
**Contracts-based Run-time Verification**

- **SoS Model with GCSL requirements**
  - Component
  - Contract
    - A: \([u > 0]\) always
    - P: \([y > 0]\) eventually
  - Monitored Component
    - Monitor
  - SoS Model enriched with Executable Monitors

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

- **Fully Automated**

- **Model Simulation**

- **Monitors Output**

- **Assumption**
  - Guarantee
SMC – A TUNABLE FORMAL ANALYSIS

Qualitative question
\[ P(\mathcal{M} \models \varphi) > ? \ p \]

Quantitative question
\[ P(\mathcal{M} \models \varphi) = ? \]

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

\( \mathcal{M} \rightarrow SMC \rightarrow \varphi \rightarrow p \rightarrow yes/no \)

Analysis results

- Precision \( \epsilon \)
- Confidence \( 1 - \delta \)

Satisfying \( \neg \varphi \)
Satisfying \( \varphi \)

Time

Simulations of \( \mathcal{M} \)

Minimal number of simulations
Analysis cost

\[ P(|\hat{p} - p| \leq \epsilon) \geq 1 - \delta \]

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

ALES S.r.l. - This page contains no technical data subject to the EAR or the ITAR - Approved for public release 28-May-15
IWTS – REQUIREMENTS ON CUSTOMER SATISFACTION

- 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

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

3. Precision = 0.05

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

Results Summary

<table>
<thead>
<tr>
<th>Precision</th>
<th>0.1</th>
<th>0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidence</td>
<td>99%</td>
<td>99%</td>
</tr>
<tr>
<td>Number of simulations</td>
<td>265</td>
<td>1060</td>
</tr>
<tr>
<td>Probability of Contract 1</td>
<td>0.86</td>
<td>0.88</td>
</tr>
<tr>
<td>Probability of Contract 2</td>
<td>0.90</td>
<td>0.89</td>
</tr>
<tr>
<td>Analysis time</td>
<td>6 min</td>
<td>25 min</td>
</tr>
</tbody>
</table>

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

DANSE Semantic Mediation-based Tool-Net

DANSE Analysis Tools

FMI

UPDM

DANSE extension

OSLC
DANSE INTEGRATED FRAMEWORK

DANSE Simulation, Analysis & Optimization Framework (SAOF)

Tool-Net SMC Platform

UPDM Rhapsody

Real SoS

GCSL Editor

Architecture Optimization Workbench

Architecture Generation

Architecture Patterns

Joint Simulation

Contract-based run-time verification

Contract Analysis

Automated Test-case Generation

Statistical Model Checking

TestCast Test Case Generation

RTANA Timing Analysis

ALES S.r.l. - This page contains no technical data subject to the EAR or the ITAR - Approved for public release

28-May-15
THANK YOU