

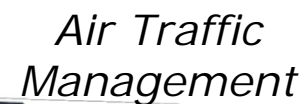
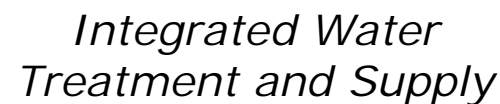
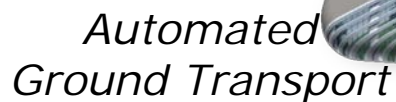


Designing for **Ad**aptability and evolution**N** in
System of systems **E**ngineering

An Effective, Tool-Supported Methodology for SoS Engineering in Europe

Near-final results from the three-year
DANSE project.

- Systems of systems concepts
- DANSE methodology
- Solution methods
- DANSE tools
- Implementation





Designing for **A**daptability and evolution**N** in
System of systems **E**ngineering

Systems of Systems Concepts

What does DANSE mean by a system of systems?

To what kinds of projects does this methodology apply?

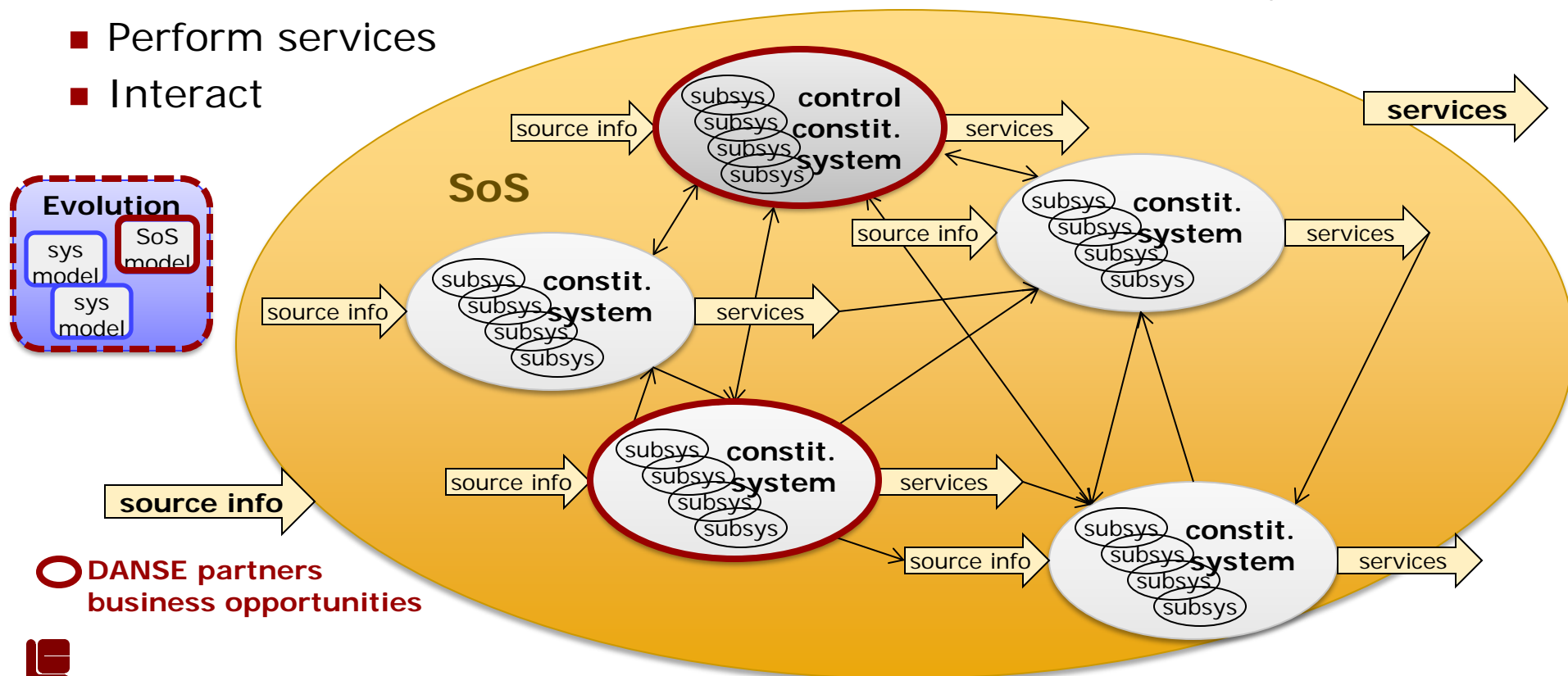
Architecture of an SoS

Constituent systems

- Independently operated and managed
- Gather/receive source info
- Perform services
- Interact

System of systems

- Provides emergent services through system interactions
- Can be modeled
- May need control



 DANSE partners
business opportunities

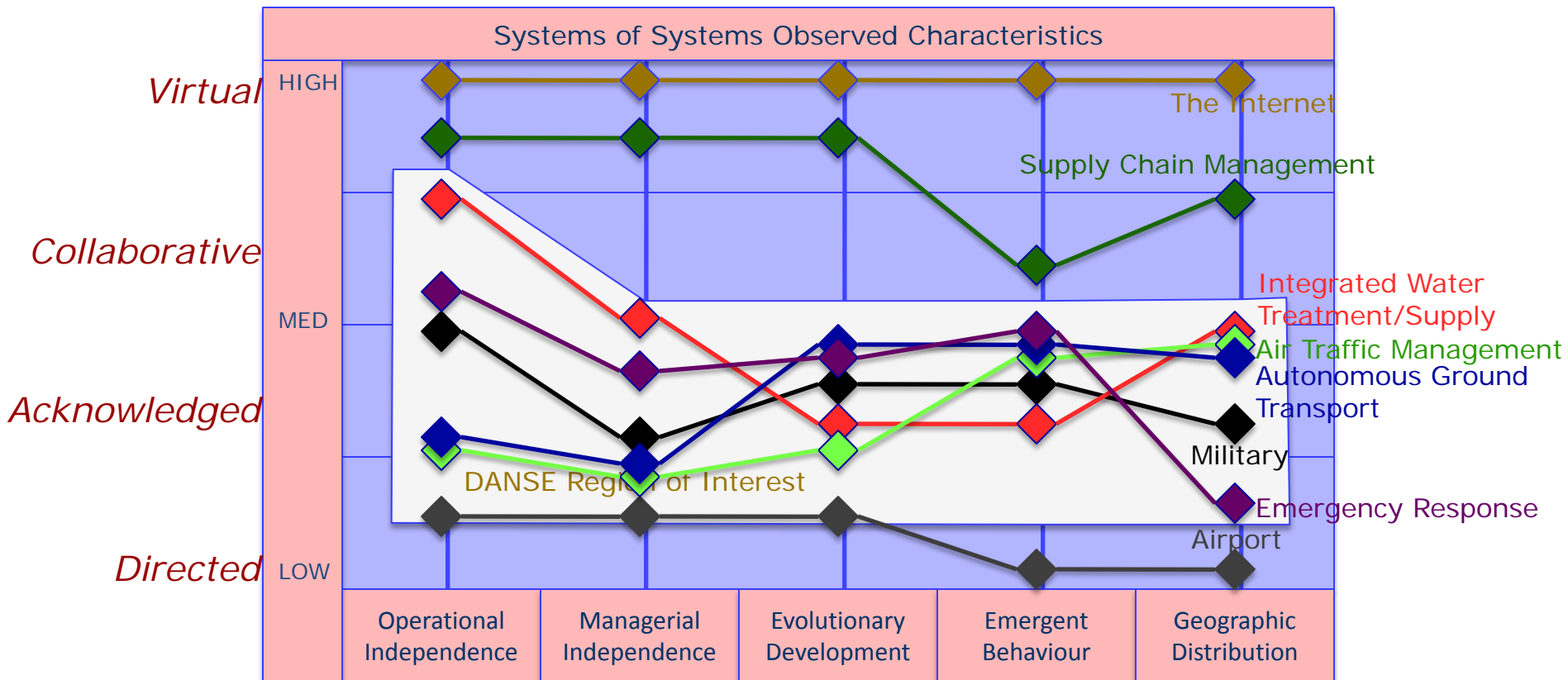
SoS Characteristics

A System is a "System of Systems" if it exhibits significant amounts of:

- **Emergent behavior** - SoS performs functions not achievable by the independent component systems
- **Geographic distribution** - geographic extent forces the elements to exchange information in a remote way
- **Evolutionary development** - functions and purposes are added, removed and modified in an ongoing way
- **Operational independence** - component systems have purpose even if detached
- **Managerial independence** - component systems are developed and managed for their own purposes

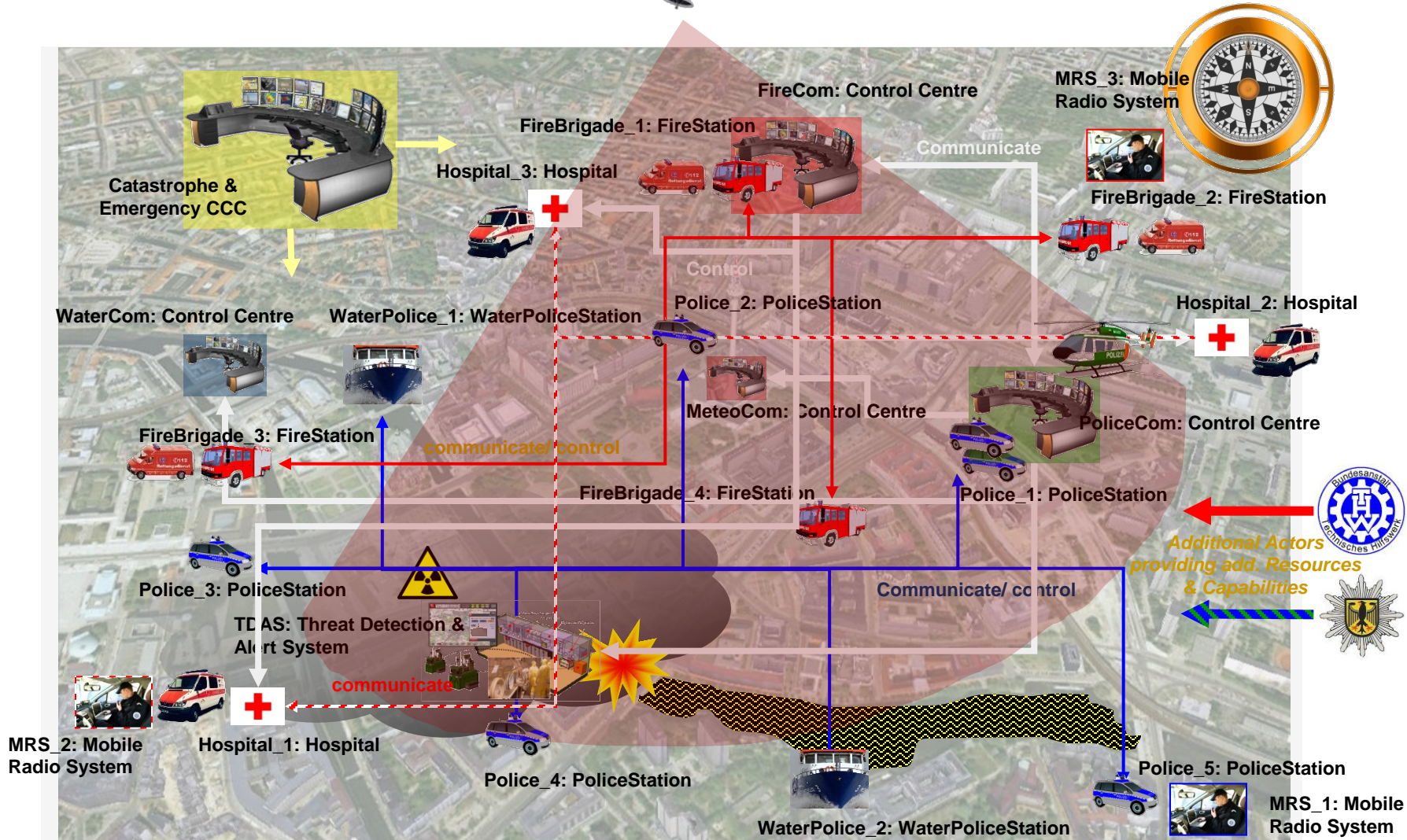
- Mark Maier 1998, "Architecting Principles for SoS," *Systems Engineering* (INCOSE)

Differing Levels of "SoS-ness"



Emergency Response SoS

Slide by Tim Lochow, EADS



Emergency Response SoS Dynamicity

Slide by Tim Lochow, EADS

SoSE Challenges

Modelling the SoS

Design Exploration Architecture Alternatives

Run Time Analysis & Simulation

Decades

Population increase

Life Cycle
Dynamicity

New fire, police and health care department stations are built or moved (More stations in order to serve smaller city areas)

Years

New buildings, roads and crossroads are created

More fire, police and health care department units are allocated

System
Dynamics

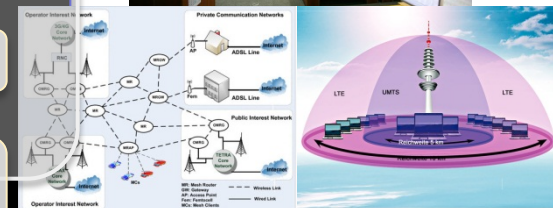
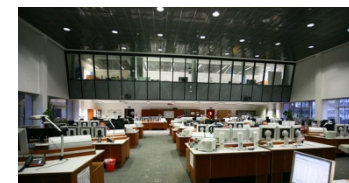
New C4I command & control organization & communication system (e.g. introduction of LTE)

Hours

Operational
Dynamicity

Improved Emergency response performance in terms of response time to emergency call and situational awareness

Minutes



SoS operational timeline and dynamicity aspects



Honour



Designing for **A**daptability and evolution**N** in
System of systems **E**ngineering

DANSE Methodology

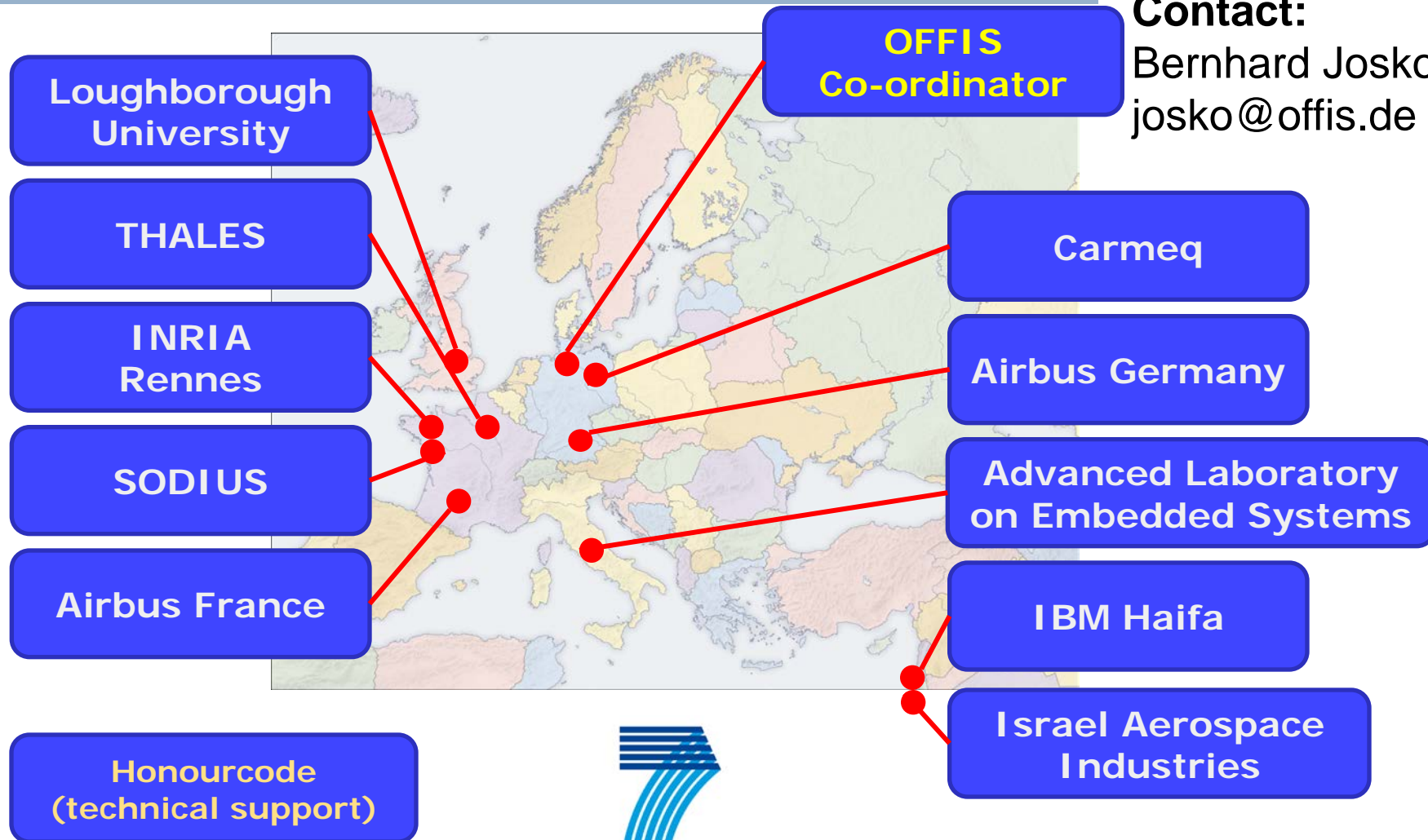
What is the DANSE project?

What is the life cycle of an SoS?

How does the DANSE methodology work in
that life cycle?

DANSE Consortium

Contact:
Bernhard Josko
josko@offis.de



DANSE Project

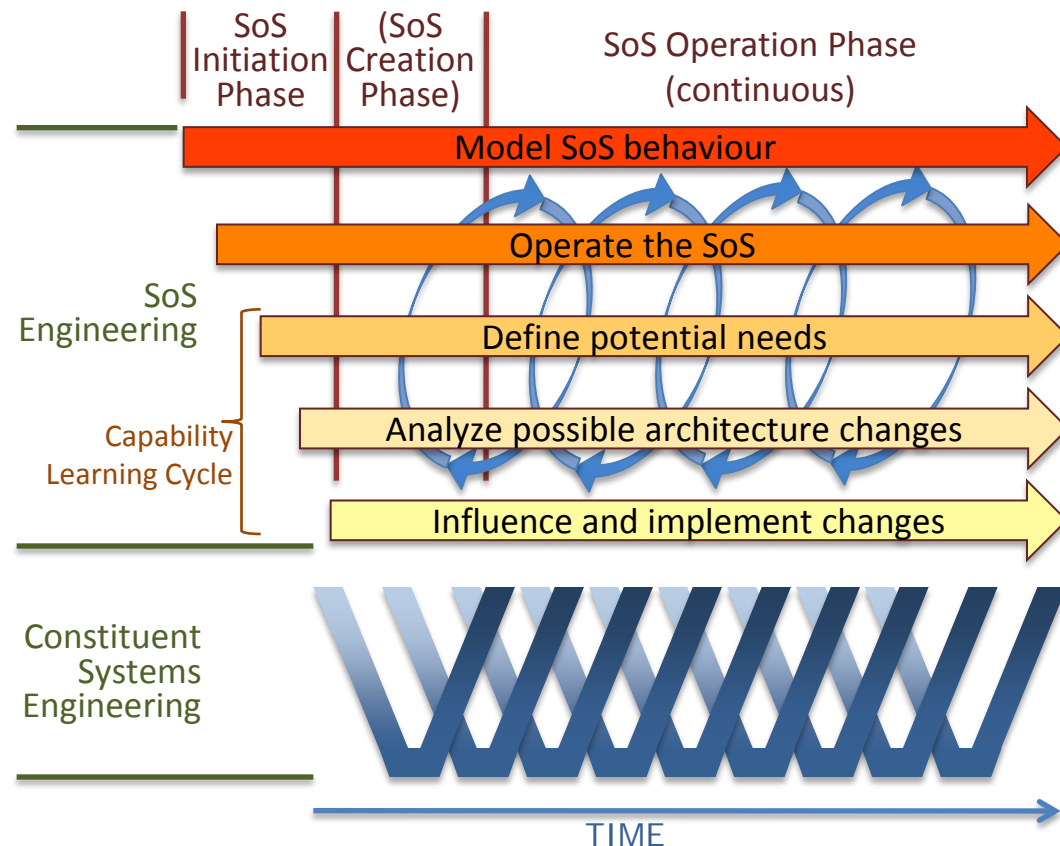


- Develop approaches for SoS engineering (design + manage)
 - Methodology to support evolution, adaptive and iterative SoS lifecycle
 - Contracts as semantically-sound model for SoS interoperations
 - Architecting Approaches for SoS – continuous and non-disruptive constituent system integration
 - Supportive tools for SoS analysis, simulation, optimization
- Validation by real-life test cases
 - Emergency Response; Integrated Water Treatment and Supply; Air Traffic Management; Autonomous Ground Transport

DANSE Methodology

Single model to embody the integrating thoughts

- An initiation phase
- Optional creation phase
- Forward movement through the SoS life
- Constant cycling of events/scenarios
- A “capability learning cycle”
 - Where the DANSE benefit happens!
- Normal Vee-based SE in the constituent systems

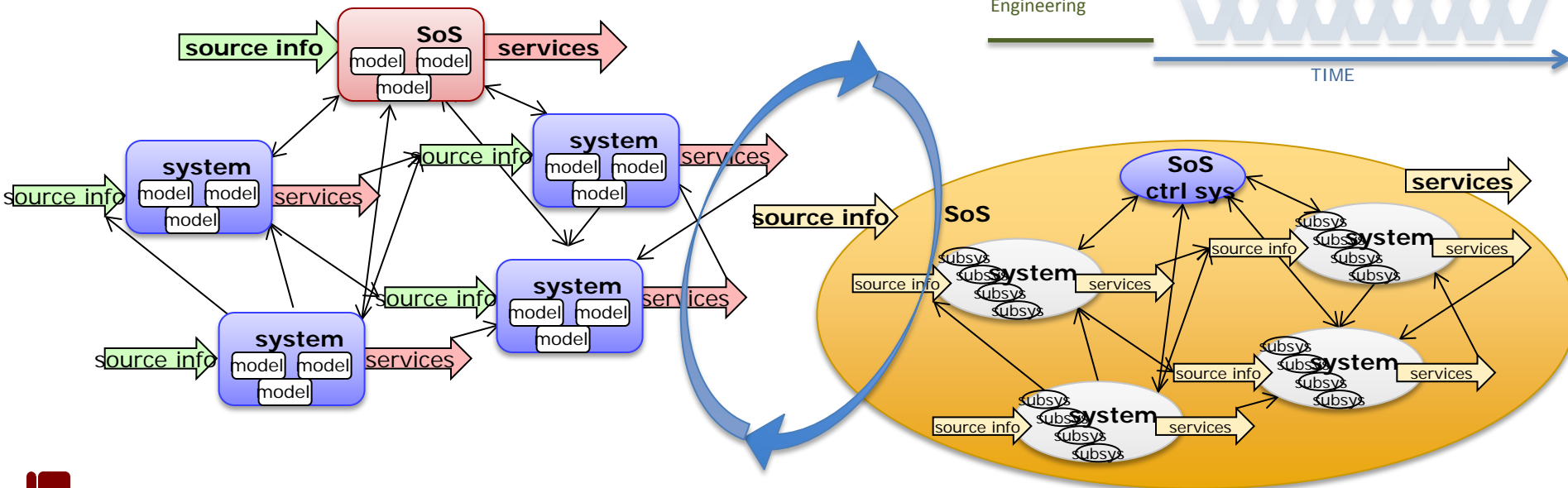
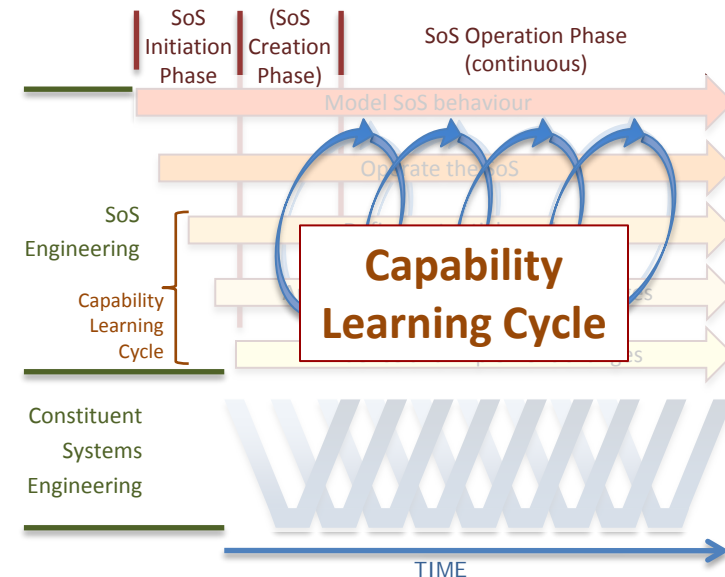


Alternate starting points:

- SoS is acknowledged among existing systems
- SoS is created by a Lead System Integrator

Capability Learning Cycle

- Constantly improve the SoS by a cycle of learning:
 - Define potential needs
 - Analyze possible architecture changes using models
 - Influence and implement changes





Designing for **A**daptability and evolution**N** in
System of systems **E**ngineering

DANSE Solution Methods

What actions can an SoS manager/architect perform within the DANSE methodology?

Solution Methods

Nbr	Solution Method	What it Does
1	Model SoS	Create UPDM SoS model, particularly focused on the SoS behaviour
2	Abstract CS model	Make a pre-existing (or new) constituent system model available for joint use with the SoS model
3	Apply architecture patterns	Build or enhance the SoS model by the use of a repository of useful patterns, proven by prior use
4	Generate architecture alternatives	Create multiple architecture alternatives for analysis, by the use of graph grammar constructs
5	Generate optimized architectures	Create and evaluate multiple architecture alternatives using concise modelling, with selection of an optimum

Solution Methods

Nbr	Solution Method	What it Does
6	Perform joint simulation	Time-based execution of a joint simulation using SoS and CS models
7	Perform statistical model checking	Identification of simulated performance levels against parameters/goals
8	Evaluate emergent behaviour	Confirmation/discovery of desired or unknown SoS emergent behaviours
9	Evaluate goals and contracts	Definition of SoS/CS goals/contracts, with automated checking during simulation
10	Perform formal verification	Knowledge of time-based compliance against formal requirements
11	Configure DANSE Tool-Net environment	Installation of necessary tools, ontologies, rules, and clients to perform DANSE modelling
12	Share models	Share SoS or CS models with other Tool-Net participants

Solution Methods in the lifecycle

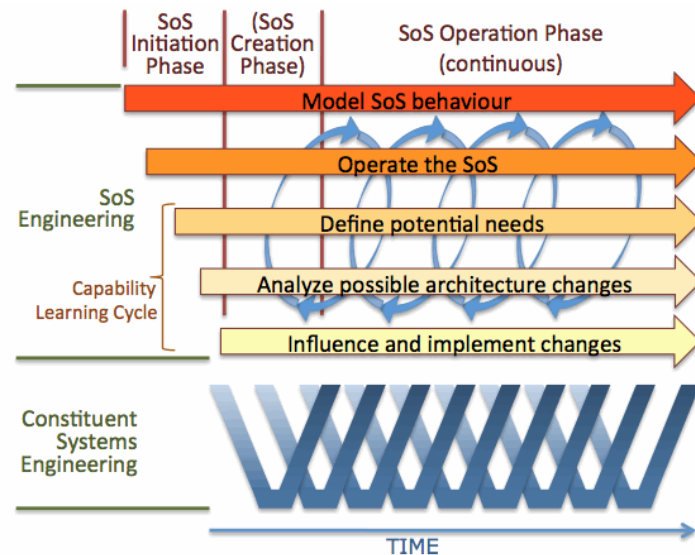
- Configure DANSE Tool-Net environment
- Model SoS
- Abstract constituent system models
- Share models

- Perform joint simulation
- Evaluate emergent behaviour

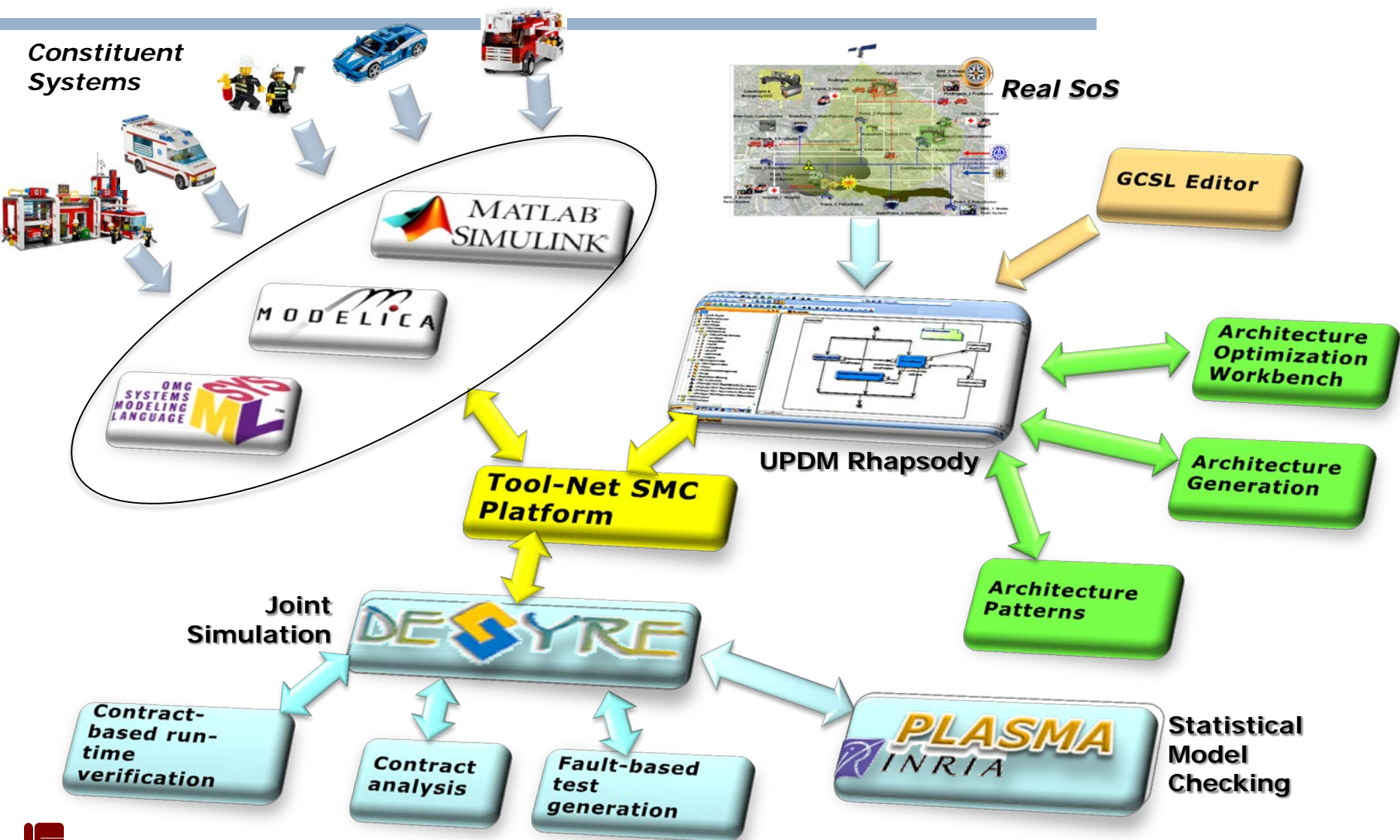
- Evaluate goals and contracts
- Perform joint simulation
- Evaluate emergent behaviour

- Evaluate goals and contracts
- Apply architecture patterns
- Generate architecture alternatives
- Generate optimized architectures
- Optimize SoS architecture
- Perform joint simulation
- Evaluate emergent behaviour
- Perform statistical model checking
- Perform formal verification
- Share models

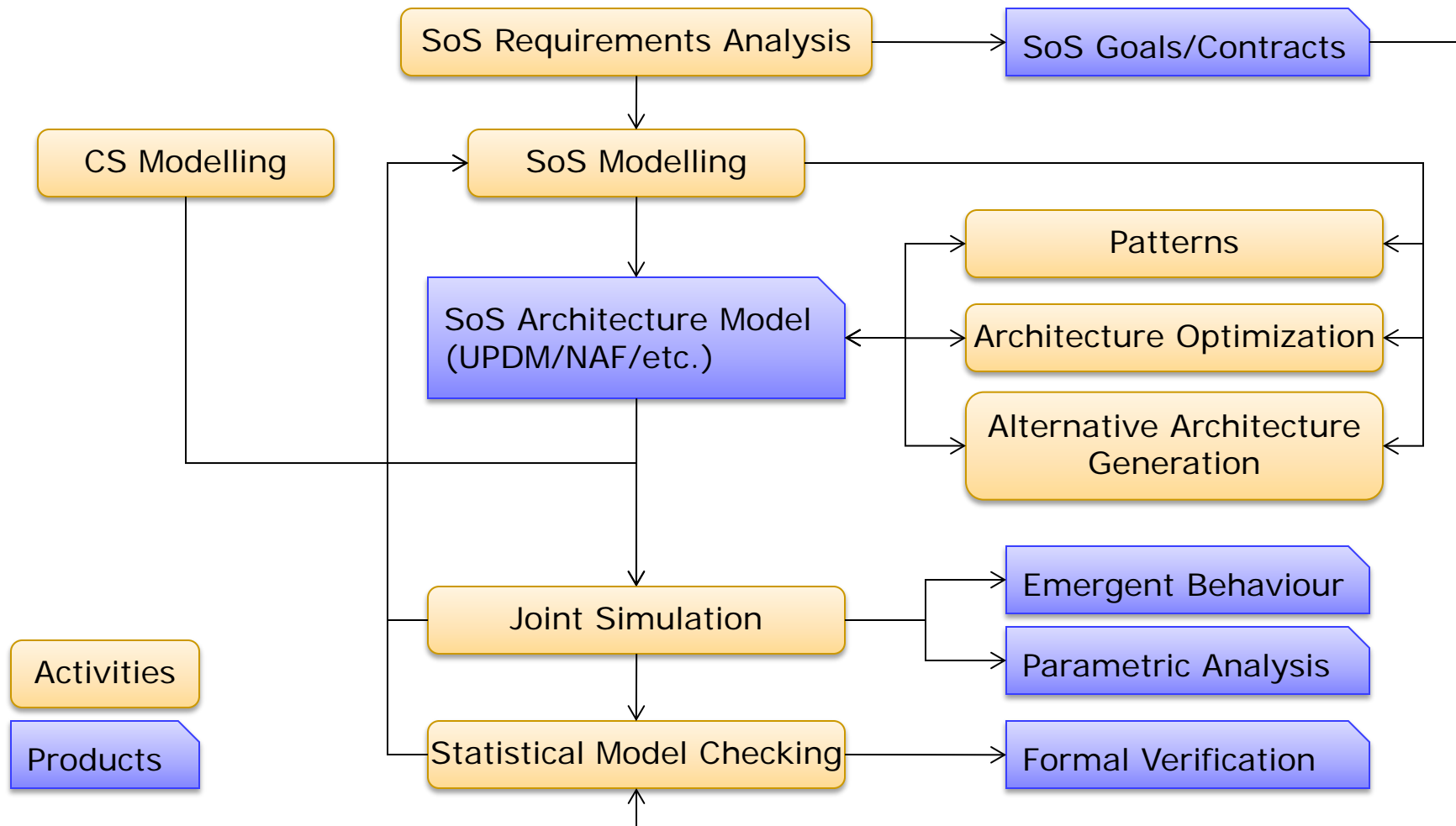
- Evaluate goals and contracts
- Optimize SoS architecture
- Perform joint simulation
- Evaluate emergent behaviour
- Perform formal verification
- Share models



DANSE Tools



Example “Use Case” of Methodology





Designing for **A**daptability and evolution**N** in
System of systems **E**ngineering

DANSE Tools

What automated tools does DANSE provide
to support the solution methods?

Solution Methods Modeling



Nbr	Solution Method	What it Does
1	Model SoS	Create UPDM SoS model, particularly focused on the SoS behaviour
2	Abstract CS model	Make a pre-existing (or new) constituent system model available for joint use with the SoS model
3	Apply architecture patterns	Build or enhance the SoS model by the use of a repository of useful patterns, proven by prior use
4	Generate architecture alternatives	Create multiple architecture alternatives for analysis, by the use of graph grammar constructs
5	Generate optimized architectures	Create and evaluate multiple architecture alternatives using concise modelling, with selection of an optimum



UPDM Overview

- "Unified Profile for DoDAF and MoDAF," also covers NAF

All Viewpoint

Capability
Viewpoint

Operational
Viewpoint

Systems
Viewpoint

Services
Viewpoint

Data &
Information
Viewpoint

Standards
Viewpoint

Project
Viewpoint

Operational Viewpoint

- OV-1 High-Level Operational Concept Graphic
- OV-2 Operational Resource Flow Description
- OV-3 Operational Resource Flow Matrix
- OV-4 Organizational Relationships Chart
- OV-5a Operational Activity Decomposition Tree
- OV-5b Operational Activity Model
- OV-6a Operational Rules Model
- OV-6b State Transition Description
- OV-6c Event-Trace Description

Systems Viewpoint

- SV-1 Systems Interface Description
- SV-2 Systems Resource Flow Descr.
- SV-3 Systems-Systems Matrix
- SV-4 Systems Functionality Description
- SV-5a Operational Activity to Systems Function Traceability Matrix
- SV-5b Operational Activity to Systems Traceability Matrix
- SV-6 Systems Resource Flow Matrix
- SV-7 Systems Measures Matrix
- SV-8 Systems Evolution Description
- SV-9 Systems Technology and Skills Forecast
- SV-10a Systems Rules Model
- SV-10b Systems State Transition Description
- SV-10c Systems Event-Trace Description

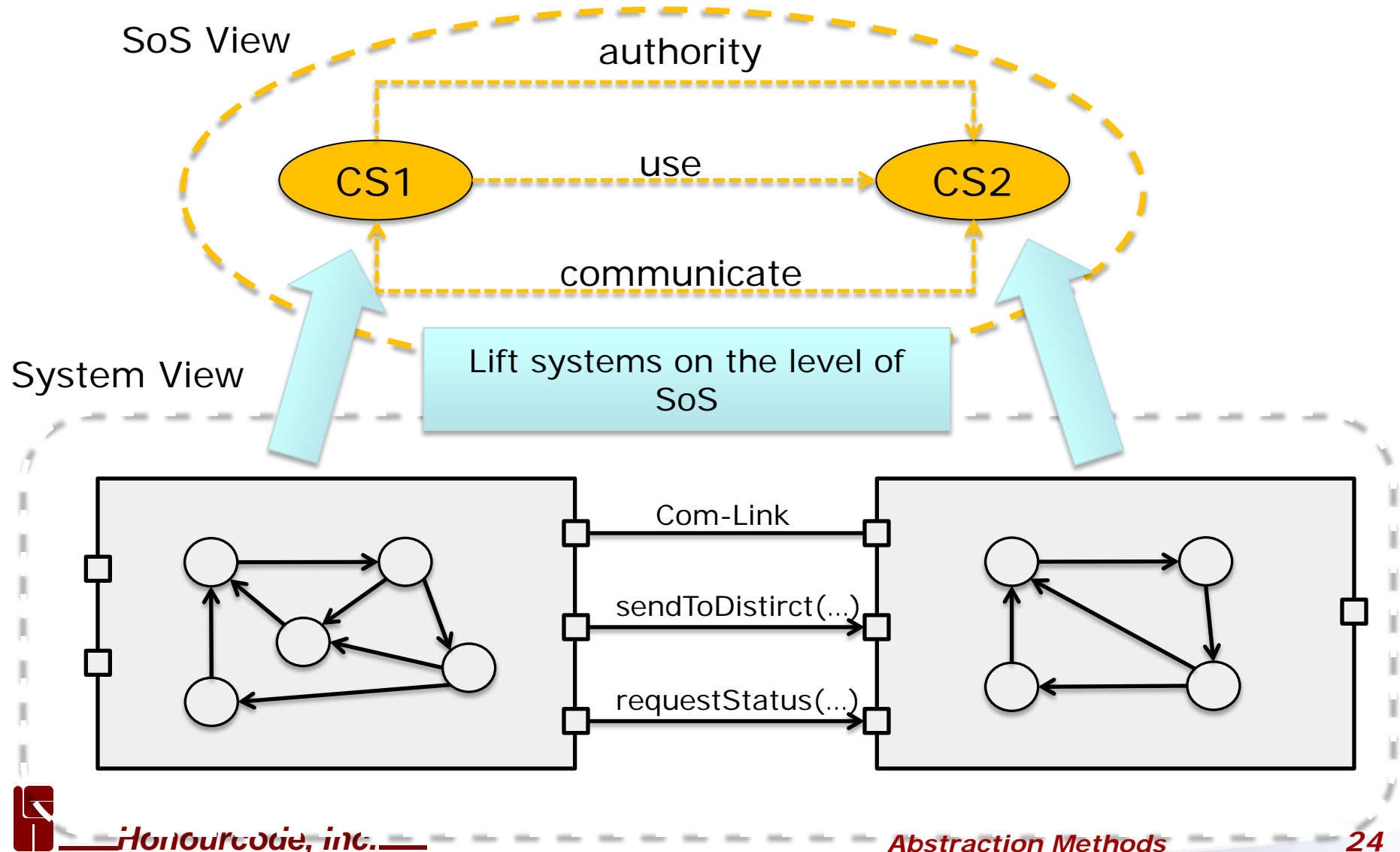


Executable UPDM Views

- SoS model should be executable as a simulation
 - Compare results with real world
 - Project “what if” scenarios
- These views support execution, lead to joint simulation

View	Name	Simulation
OV-5a	Operational Activity Decomposition Tree	Structure of OV executable elements
OV-5b	Operational Activity Model	SysML Use Case, Activity forms
OV-6b	State Transition Description	SysML State diagrams
OV-6c	Event-Trace Description	SysML Sequence diagrams
SV-1	Systems Interface Description	Structure of SV executable elements
SV-4	Systems Functionality Description	SysML Use Case, Activity forms
SV-10b	Systems State Transition Description	SysML State diagrams
SV-10c	Systems Event-Trace Description	SysML Sequence diagrams

Constituent System Models



Abstraction Methods

Partner

Group nodes with similar interactions

Flow

Focus on I/O and key parameters

Spotlight

Focus on key elements, others generalized

Steady State

Focus on stable states & transitions among

Statistical

Match statistical behavior w/o details

Timing

Focus on timing issues; other issues ignored

...others also exist

Solution Methods

Goals and Contracts



Nbr	Solution Method	What it Does
6	Perform joint simulation	Time-based execution of a joint simulation using SoS and CS models
7	Perform statistical model checking	Identification of simulated performance levels against parameters/goals
8	Evaluate emergent behaviour	Confirmation/discovery of desired or unknown SoS emergent behaviours
9	Evaluate goals and contracts	Definition of SoS/CS goals/contracts, with automated checking during simulation
10	Perform formal verification	Knowledge of time-based compliance against formal requirements
11	Configure DANSE Tool-Net environment	Installation of necessary tools, ontologies, rules, and clients to perform DANSE modelling
12	Share models	Share SoS or CS models with other Tool-Net participants



Goals and Contracts as SoS "Requirements"

- Goal: statement of a desired condition, with quantifiable measurement of the degree to which it is met.
 - *Communications coverage over the urban area (% area covered)*
 - *Response time to a fire (minutes between call and arrival)*
 - *Graded levels of performance*
- Contract: statement of an essential condition, with quantifiable measurement whether it is met.
 - *Within the urban area, response time to a fire is no greater than 15 minutes with probability 99.5%*
 - *Catastrophe and Emergency Center has direct communications with all fire, police, ambulance centers*
 - *Yes/No evaluation*



Goals/Contracts Specification Language (GCSL) Overview



- Bridges the gap between
 - Natural language used by people
 - Formal languages required by analysis tools
- Textual pattern with specific semantics
- Formalization process
 1. Define natural language goals/contracts
 2. Structure each statement into the
 - Assumption part ("If X is true...") and the
 - Promise part ("...then Y must be true")
 3. Select a GCSL pattern for the type of relation
 4. Write "X" and "Y" in the GCSL syntax

Catastrophe and Emergency Center has direct comms with all police centers



*If CEC exists...
...then it has direct comms with all police centers*

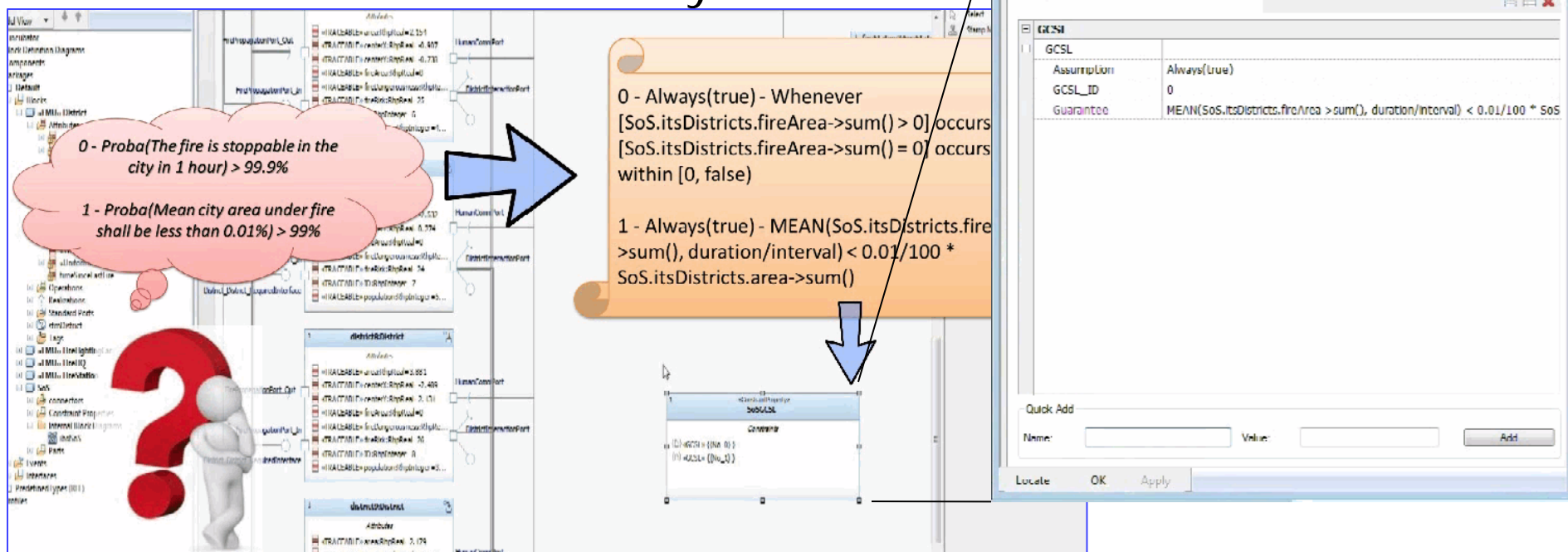


```
SoS.itsCEC->exists(CEC) implies  
SoS.itsCEC->ForAll(PoliceCenter->comms=true)
```



GCSL Editor

- Rhapsody plug-in, part of DANSE UPDM profile extensions
- Create UPDM block
 - Associated with SoS object of interest
 - Contains GCSL statements
- GCSL Editor checks syntax



The screenshot shows the GCSL Editor interface. On the left, a SoS diagram is visible with a red question mark icon. A pink cloud contains two GCSL statements:

- 0 - $\text{Proba}(\text{The fire is stoppable in the city in 1 hour}) > 99.9\%$
- 1 - $\text{Proba}(\text{Mean city area under fire shall be less than } 0.01\%) > 99\%$

A blue arrow points from the cloud to a yellow box containing the formal GCSL statements:

- 0 - Always(true) - Whenever $[\text{SoS.itsDistricts.fireArea} \rightarrow \text{sum}() > 0]$ occurs $[\text{SoS.itsDistricts.fireArea} \rightarrow \text{sum}() = 0]$ occurs within $[0, \text{false}]$
- 1 - Always(true) - $\text{MEAN}(\text{SoS.itsDistricts.fire} \rightarrow \text{sum}(), \text{duration}/\text{interval}) < 0.01/100 * \text{SoS.itsDistricts.area} \rightarrow \text{sum}()$

A blue arrow points from the yellow box to a GCSL block in the diagram. The block is labeled "GCSL" and contains the following text:

```

(0) <GCSL> = {true, 0}
(1) <GCSL> = {true, 0}
    
```

On the right, the GCSL Editor window is open, showing the "General" tab. It displays the GCSL statement editor with the following fields:

Field	Value
Assumption	Always(true)
GCSL_ID	0
Guarantee	$\text{MEAN}(\text{SoS.itsDistricts.fire} \rightarrow \text{sum}(), \text{duration}/\text{interval}) < 0.01/100 * \text{SoS.itsDistricts.area} \rightarrow \text{sum}()$

At the bottom of the window, there is a "Quick Add" section with fields for "Name" and "Value", and an "Add" button.

Solution Methods

Architecture Exploration

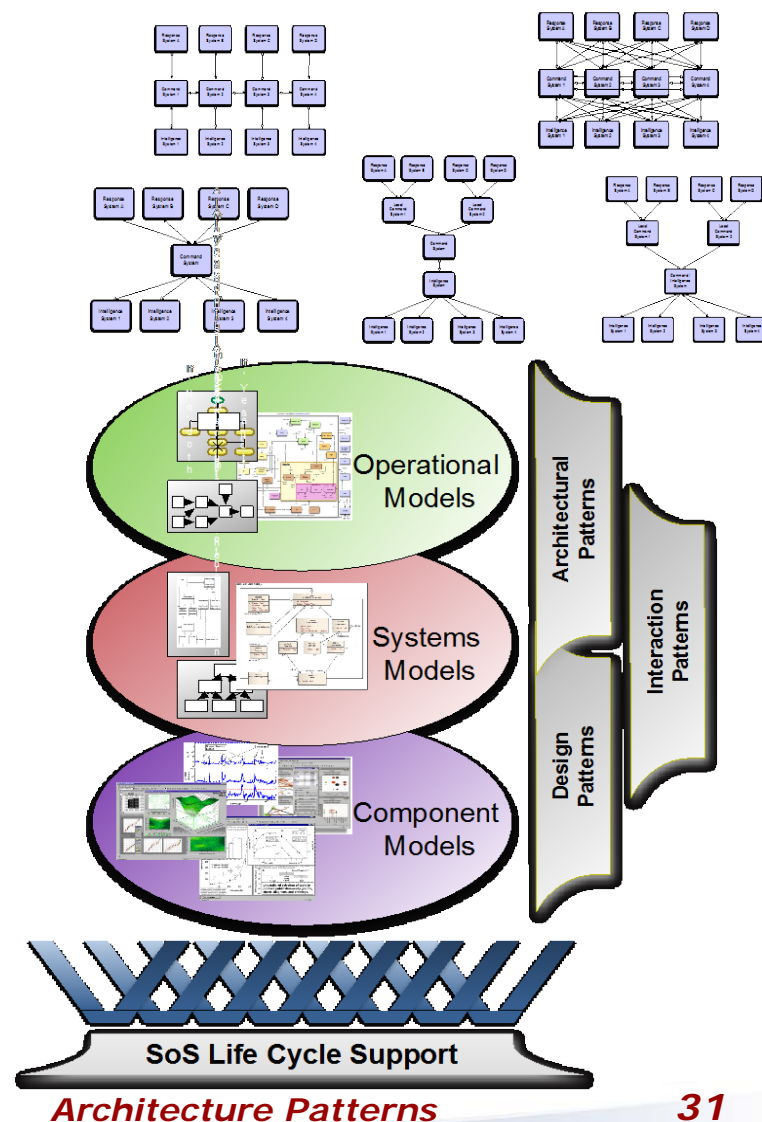


Nbr	Solution Method	What it Does
1	Model SoS	Create UPDM SoS model, particularly focused on the SoS behaviour
2	Abstract CS model	Make a pre-existing (or new) constituent system model available for joint use with the SoS model
3	Apply architecture patterns	Build or enhance the SoS model by the use of a repository of useful patterns, proven by prior use
4	Generate architecture alternatives	Create multiple architecture alternatives for analysis, by the use of graph grammar constructs
5	Generate optimized architectures	Create and evaluate multiple architecture alternatives using concise modelling, with selection of an optimum



SoS Architecture Patterns

- Templates to describe solutions to known problems
 - **Context – Problem - Solution**
- Provide a generalized guideline to realize certain architecture characteristics.
- Built on a common anatomy
- DANSE has developed an SoS pattern repository
 - Searchable database of patterns
 - UPDM profiles that can be inserted into the SoS model



Architecture Pattern Anatomy

Any key words that may appear in the pattern that will be useful when looking up the pattern in a repository.

The Author of the Pattern

This refers to the problem and why you would use the pattern to address the issue.

Also known as.



Statement of why the pattern would be utilised to address the design problem or situation. It will help understand the structure and consequences later in the pattern.

Name of Pattern

ARCHITECTURE PATTERNS REPOSITORY

Command, Control and Execution Architecture Patterns

Go to Printing Layout

Catalogue Previous Next All Patterns

Pattern Name

Centralised Command and Control Pattern

Keywords

Command, Centralised, Control

Author

Loughborough Uni

☐ New Pattern

Pattern Information

Intent

Exercise of authority (invention, advice, opinion, influence, or command) and direction by a control system over assigned resources to achieve accomplishment of the specified mission. The Central Command/Control System governs and exercises full authority over resources.

Also Known

CCC, C3

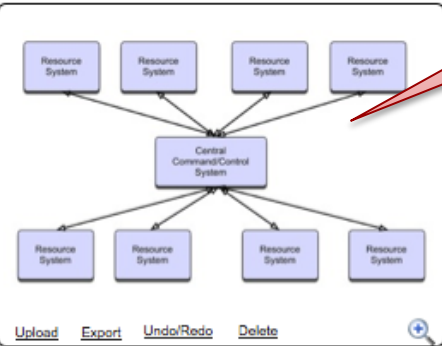
Motivation: Goals

The pattern allows for a single command centre, which has unquestionable overall authority.

Motivation: Capabilities

1. Many points of intelligence access the Centralised Command System, allowing for a more informed decision making process.
2. Centralised command facilitates for all knowledge to be in one central location, resulting in more accurate decision making.

Using the Pattern



Structure

Applicability

Root Architecture Patterns

Model (SysML)

SoSArchitecturePattern Export Delete Upload

Model (SysML+Concise)

Upload

Model (UPDM)

Upload

Model (UPDM+Concise)

Upload

Model (Other)

Upload

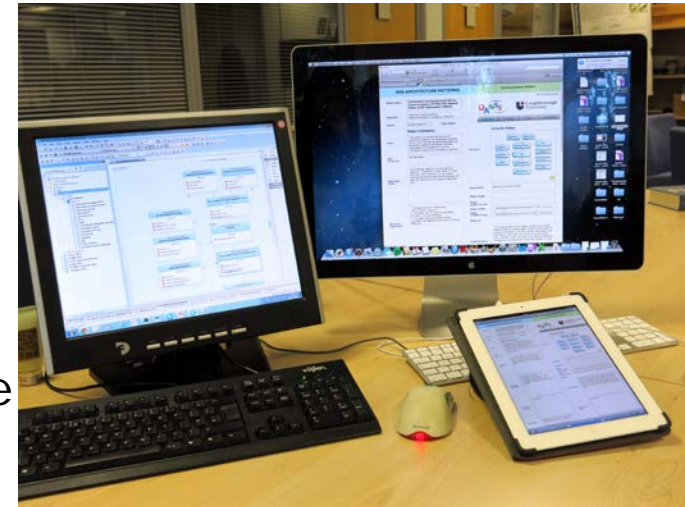
Diagram of Pattern's Structure

Rhapsody Models Available for Download

14 other fields also available

Architecture Patterns Repository

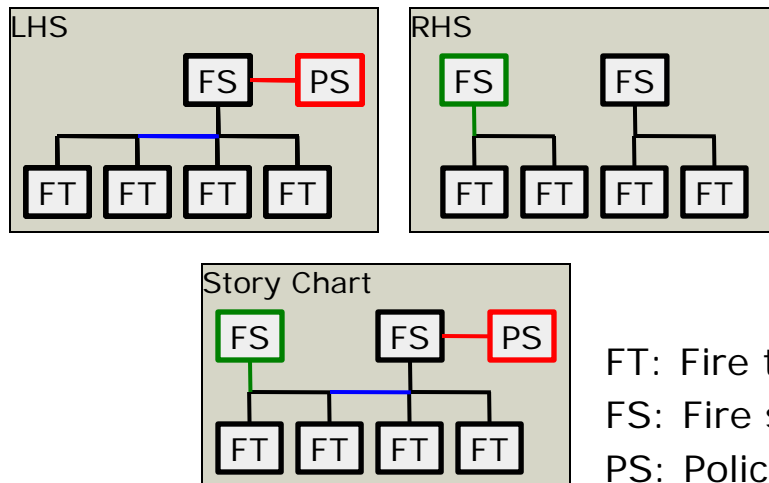
- Architecture Patterns repository includes larger catalog of patterns
 - e.g. UPDM, SysML, Test Cases etc.
- Sophisticated online repository for architecture patterns with powerful search capabilities, option to store new patterns.
- The repository exists itself in three forms;
 - a document-based repository,
 - a repository of IBM Rhapsody profiles, and
 - an online searchable repository with the option to download IBM Rhapsody SysML/UPDM profiles for inclusion in DANSE Tool-net.
- Accessed via:
 - Conventional web browser (all popular browsers supported),
 - Apple iPad running the free FileMaker App – FileMaker Go.
 - User run-time version of FileMaker



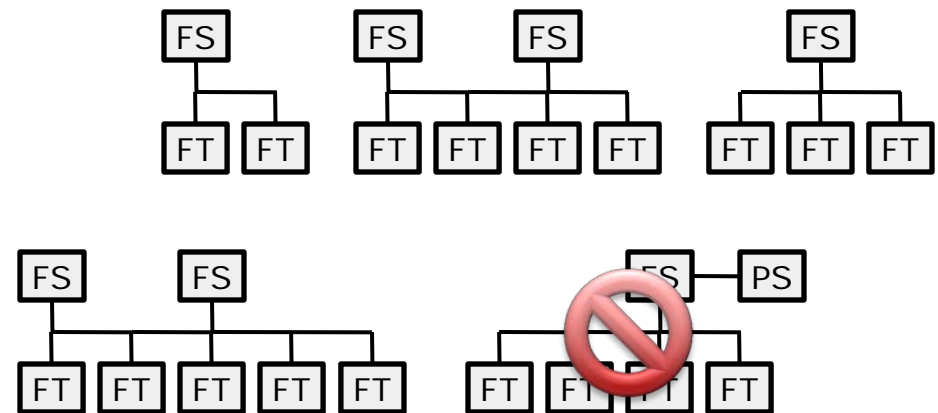
Graph Grammar

- 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*

- Reader:** Matched, not changed.
- Eraser:** Matched and **removed**.
- Creator:** **Added** to the model.
- Embargo:** Prevents the match.

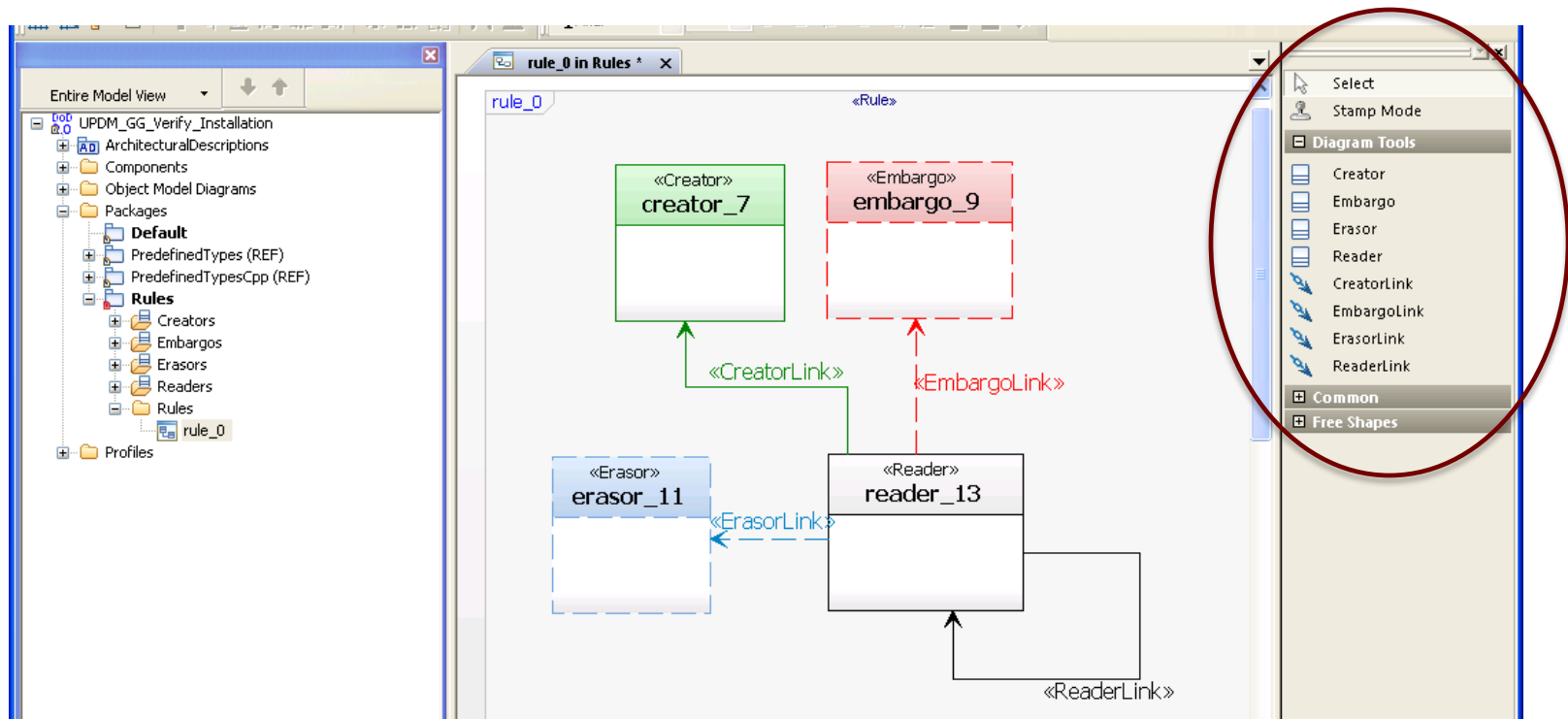


FT: Fire truck
FS: Fire station
PS: Police station

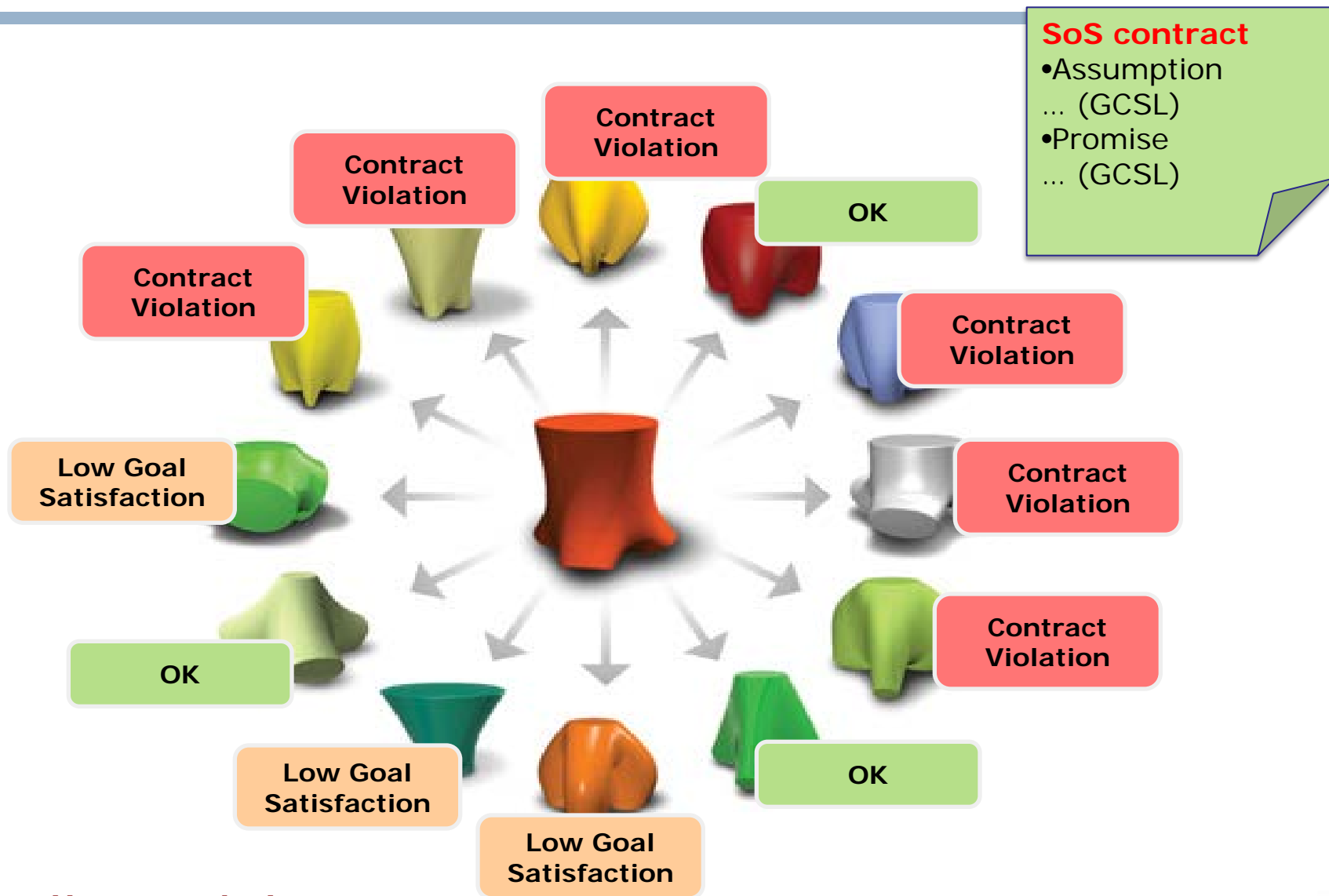


DANSE Graph Grammar

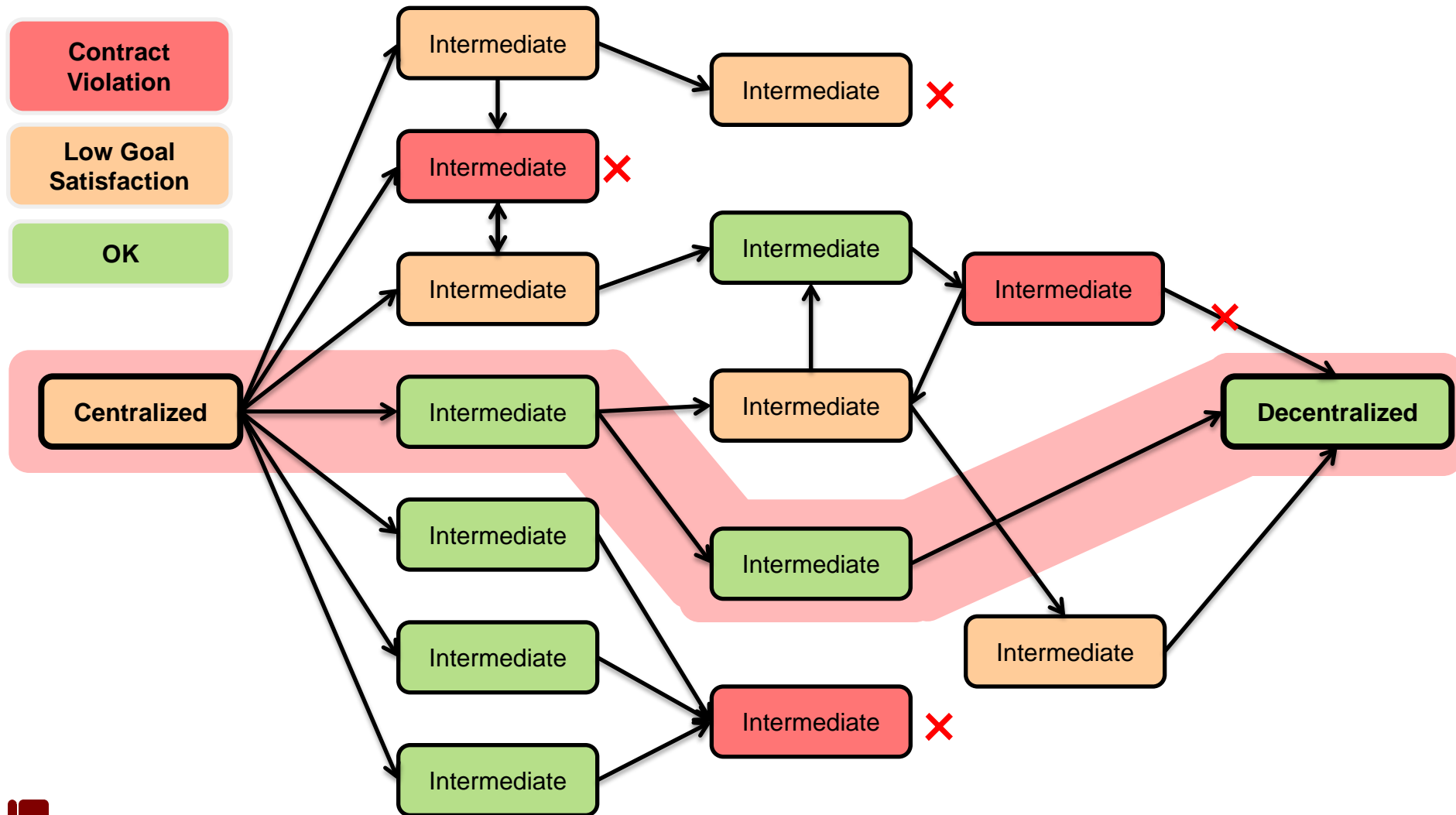
- Story Charts **implemented** as special UPDM diagrams
- Based on a UPDM **profile** to enable the modeling of a rule
- Revised models created automatically by applying the rules



Exploration of Design Space



Reachability of Future Architectures



Architecture Optimization Concept

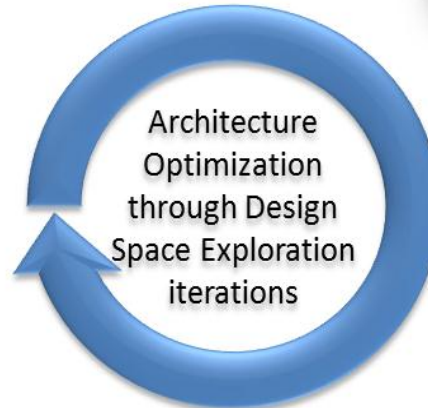
1. Describe system through different SysML views, including design alternatives, constraints and goals

2. Derived Data Schema for Input and output structures

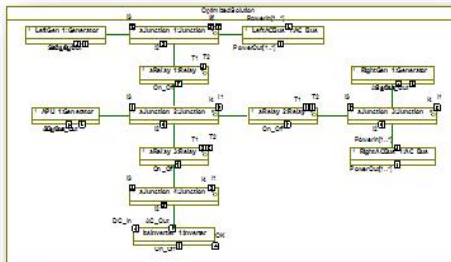
idname	guid	variants
1JunctionGUID	e1eb1581-c487-4be2-9fd6-8ce56a5b2ea	Cat.Junction
3Relay	GUID 8feb223d-5bc6-40d6-b179-46b5ff3d7e58	Cat.Relay

id	name	guid	typed type name	variants	date from to
23Pacemaker	GUID 39e335-5e09-4801-898a-e53ae34332e2	23Pacemaker	cu.Pacemaker		
18PigGen	GUID 388e77a7-5b4d-48a0-888a-f63a680a0c72	18PigGen	cu.PigGen		
1315Pacemaker	GUID 8c0c97f7-08e4-4a08-0743-30e73687336c	1315Pacemaker	cu.Pacemaker		
18PigGenBus	GUID 6b34f0e5b4e-07f7-40b3-0d9f6187e9e7	18PigGenBus	18AC_Bus		
24PU	GUID 14d13109-f4e402-0932-0e40b3533333	24PU	cu.PU		
1515PigGen	GUID 120a669-1109-40ca-e82e-79d37e6a0042	1515PigGen	cu.PigGen		
2415Pacemaker	GUID 2150603-e0d8-07f7-0e2-44a831a734	2415Pacemaker	cu.Pacemaker		
2415Pacemaker	GUID 3a20f00-f81a-4a09-910b-03321a634c1	2415Pacemaker	cu.Pacemaker		
1515PigGen	GUID 8813a39-f5d8-07f7-1802b-6051512	1515PigGen	cu.PigGen		
1415PacBus	GUID 86a0e7-0068-44a0-0032-073247e2808f	1415PacBus	18AC_Bus		
1515Pacemaker	GUID 8c7650f7031-402a-4a36-c33-031a354e	1515Pacemaker	cu.Pacemaker		

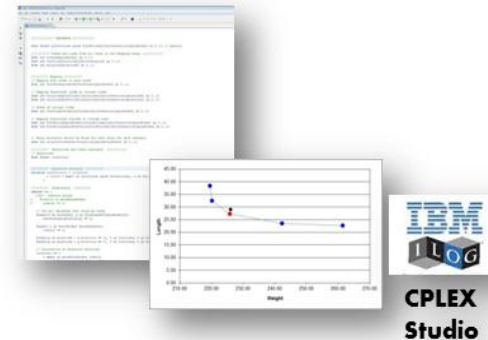
Rational software
Rhapsody



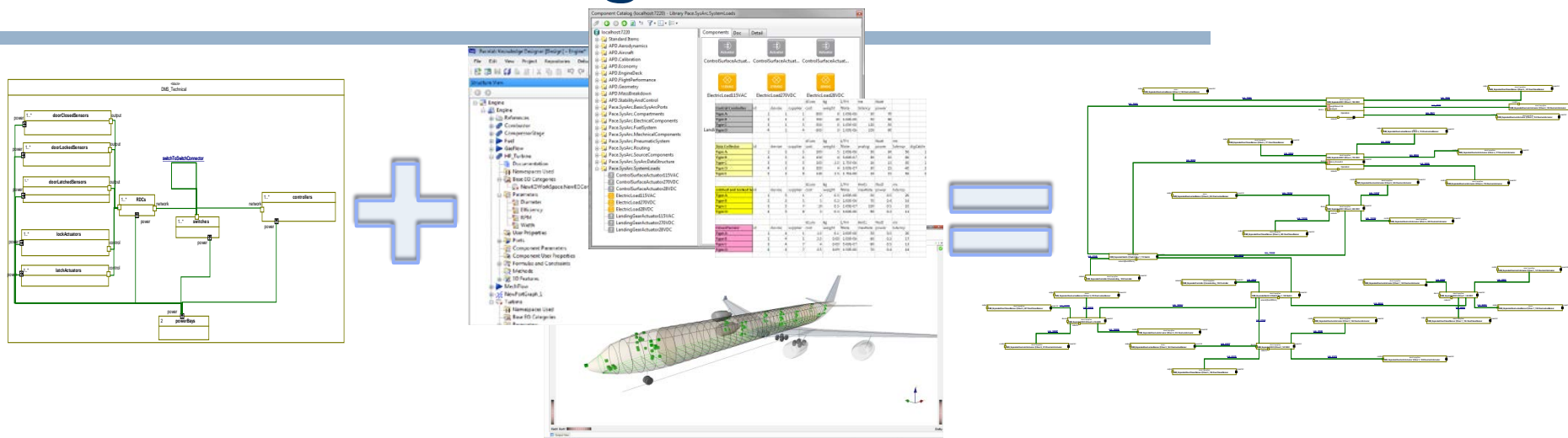
4. Optimized architecture back annotated to SysML model



3. Automatic translation(via an interchange format) into Optimization solver



Concise Modeling



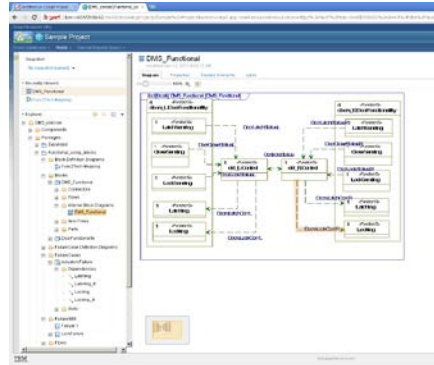
SysML models combined with tabular data

- SysML depicts the system composition rules (architectural template or pattern)
- Tables contain instantiations, variations in quantities or parameters
- Automatic Generation tool creates architecture variants by applying the table data to the template

Dashboard for Architecture Optimization



Modeling



Data management

Parameter	Value	Unit	Category	Sub-category
Weight	100	kg	Mass	Weight
Length	100	m	Length	Length
Area	100	m²	Area	Area
Volume	100	m³	Volume	Volume
Power	100	W	Power	Power
Cost	100	\$	Cost	Cost
Energy	100	J	Energy	Energy
Time	100	s	Time	Time
Frequency	100	Hz	Frequency	Frequency
Temperature	100	°C	Temperature	Temperature
Pressure	100	Pa	Pressure	Pressure
Force	100	N	Force	Force
Moment	100	Nm	Moment	Moment
Acceleration	100	m/s²	Acceleration	Acceleration
Velocity	100	m/s	Velocity	Velocity
Displacement	100	m	Displacement	Displacement
Angle	100	rad	Angle	Angle
Angular Velocity	100	rad/s	Angular Velocity	Angular Velocity
Angular Acceleration	100	rad/s²	Angular Acceleration	Angular Acceleration
Rotational Inertia	100	kgm²	Rotational Inertia	Rotational Inertia
Rotational Velocity	100	rad/s	Rotational Velocity	Rotational Velocity
Rotational Acceleration	100	rad/s²	Rotational Acceleration	Rotational Acceleration
Rotational Inertia	100	kgm²	Rotational Inertia	Rotational Inertia
Rotational Velocity	100	rad/s	Rotational Velocity	Rotational Velocity
Rotational Acceleration	100	rad/s²	Rotational Acceleration	Rotational Acceleration

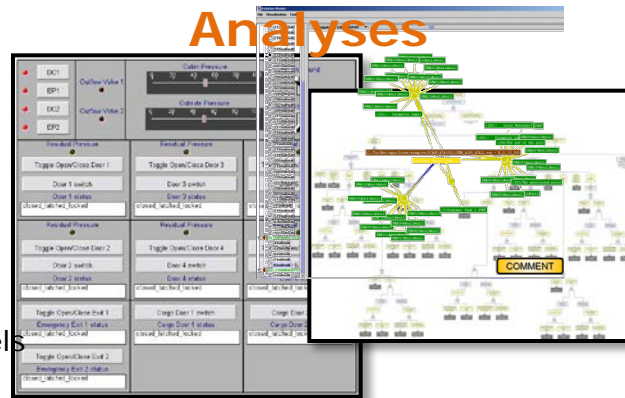
Objectives management

Objective	Expression	Priority	Weight
Weight	minimize	1	100
Length	minimize	1	100
Area	minimize	1	100
Volume	minimize	1	100
Power	minimize	1	100
Cost	minimize	1	100
Energy	minimize	1	100
Time	minimize	1	100
Frequency	minimize	1	100
Temperature	minimize	1	100
Pressure	minimize	1	100
Force	minimize	1	100
Moment	minimize	1	100
Acceleration	minimize	1	100
Velocity	minimize	1	100
Displacement	minimize	1	100
Angle	minimize	1	100
Angular Velocity	minimize	1	100
Angular Acceleration	minimize	1	100
Rotational Inertia	minimize	1	100
Rotational Velocity	minimize	1	100
Rotational Acceleration	minimize	1	100

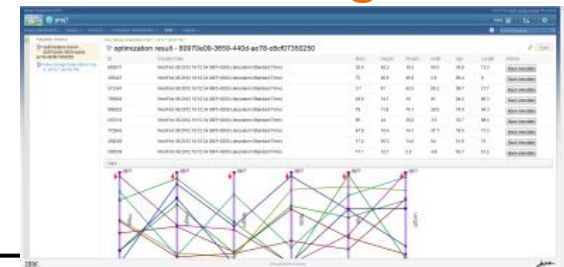
- Single environment
 - Main tool of the Systems Engineer
 - Controlling the design and analysis process
- Based on Design Manager and JTS
- Interaction with modeling environments
 - Review and comment mechanisms
 - Models import / export control
 - Back-end model transformations
- Integration with analysis tools
 - Simulations, computations, domain specific views
 - White-box, black-box
 - Analysis results feedback into models
- Visual analytics



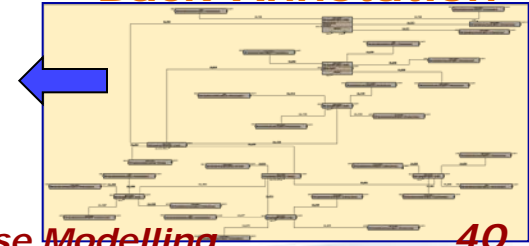
Multiple Analyses



Results management



Back-Annotation

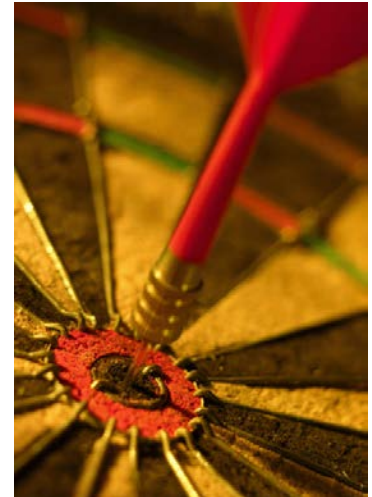


Joint Simulation and Analysis

Nbr	Solution Method	What it Does
6	Perform joint simulation	Time-based execution of a joint simulation using SoS and CS models
7	Perform statistical model checking	Identification of simulated performance levels against parameters/goals
8	Evaluate emergent behaviour	Confirmation/discovery of desired or unknown SoS emergent behaviours
9	Evaluate goals and contracts	Definition of SoS/CS goals/contracts, with automated checking during simulation
10	Perform formal verification	Knowledge of time-based compliance against formal requirements
11	Configure DANSE Tool-Net environment	Installation of necessary tools, ontologies, rules, and clients to perform DANSE modelling
12	Share models	Share SoS or CS models with other Tool-Net participants

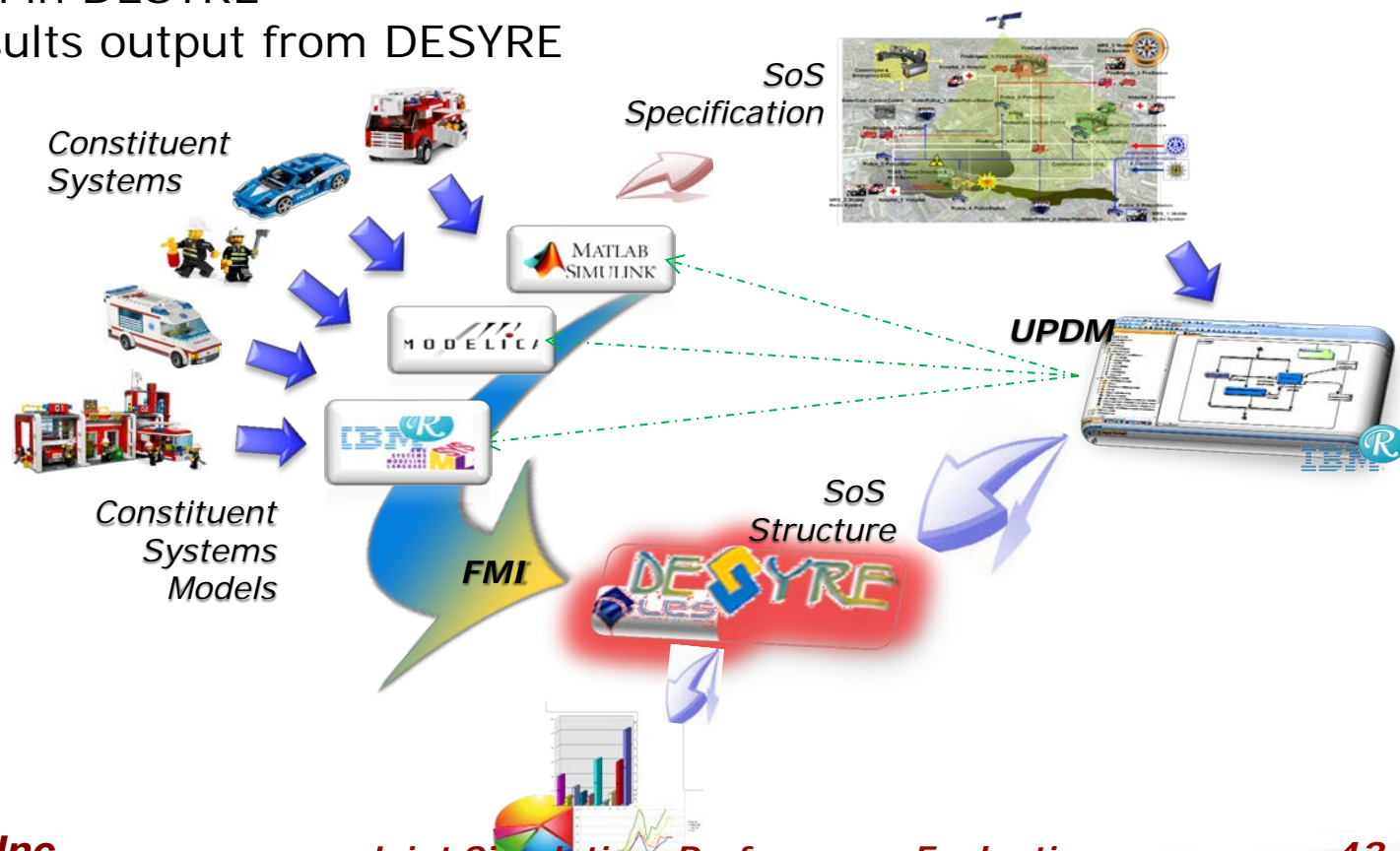
Performance Evaluation Concepts

- Have generated multiple architecture alternatives
 - Patterns application
 - Graph grammar automated generation
 - Concise modeling with optimization
- Need SoS joint simulation to evaluate performance
 - Predict characteristics of interest
 - Evaluate contracts and goals during simulation
 - Dynamic aspects of optimization
 - Stochastic variability
- Provide information for decision analysis



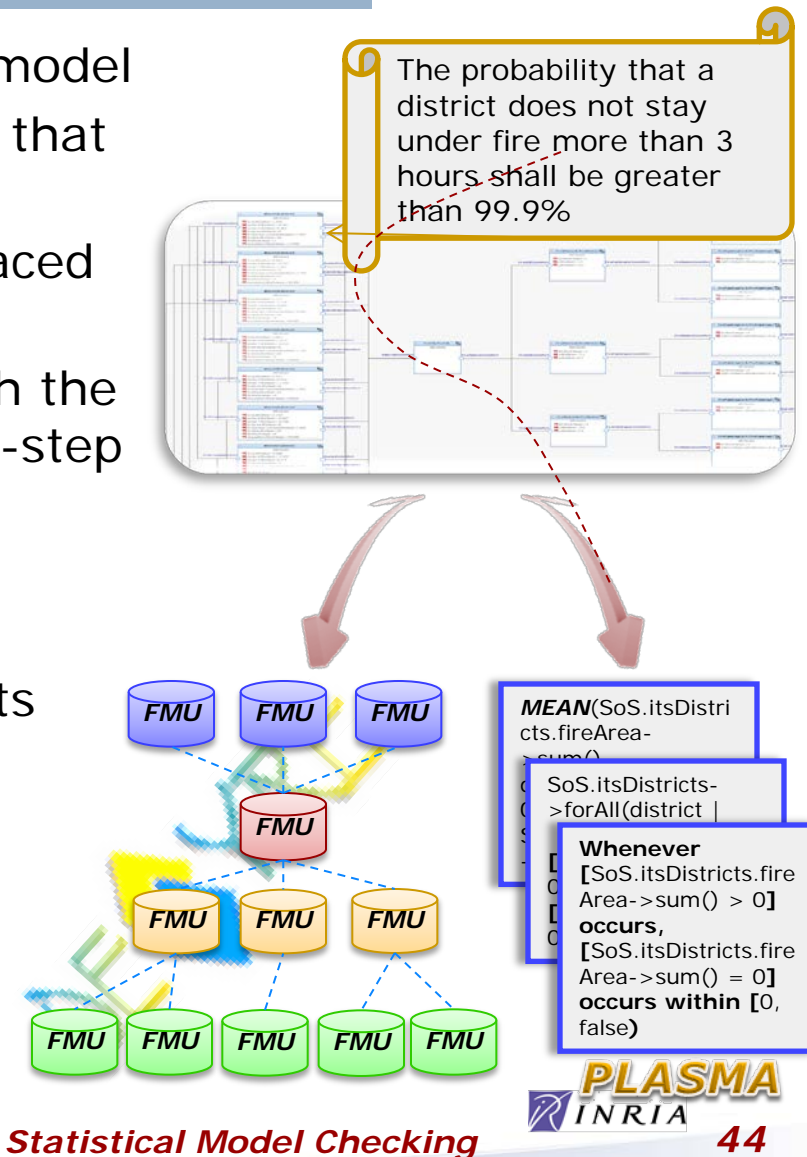
Joint simulation

- FMI standard for component integration
- Constituent system models exported as FMUs from tool
- SoS architecture exported to DESYRE
- FMUs imported in DESYRE
- Simulation run in DESYRE
- Simulation results output from DESYRE



Statistical Model Checking

- Goals and Contracts specified in UPDM model
- GCSL translated into a set of properties that can be evaluated by PLASMA
- UPDM parameters set as observable, traced by the simulator
- DESYRE simulator provides PLASMA with the value assumed by the variables step-by-step during the simulation
- PLASMA verifies the properties
- PLASMA returns the Statistical Model Checking and contract verification results





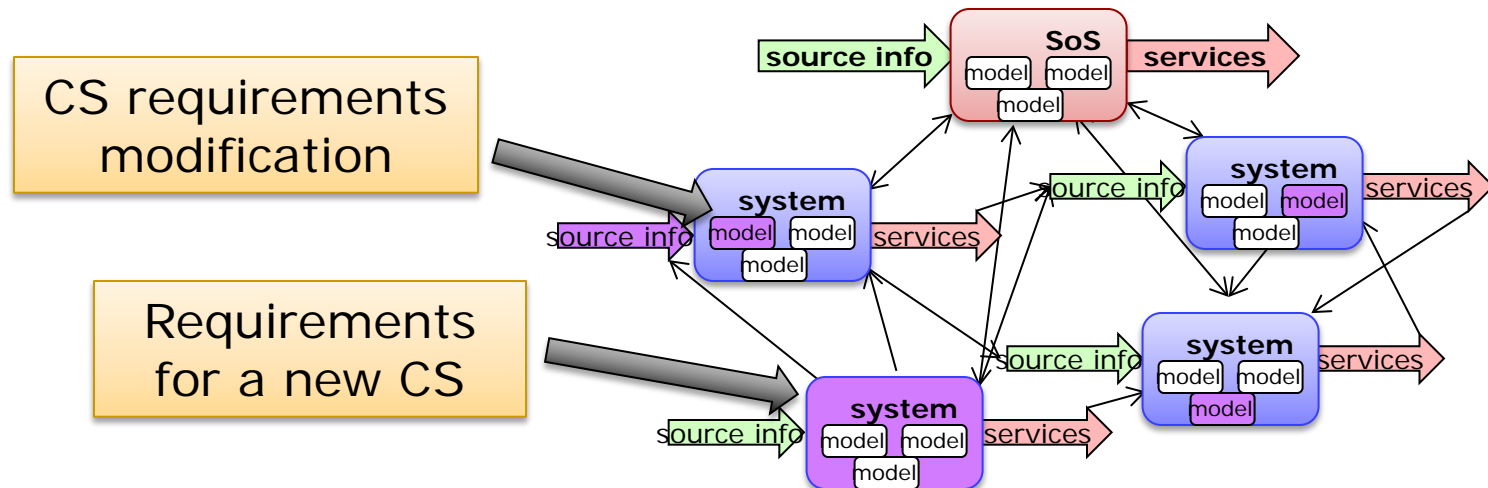
Designing for **A**daptability and evolution**N** in
System of systems **E**ngineering

Implementation

How does the DANSE methodology support change in the SoS?

Constituent System Requirements

- Each change to the SoS and constituent system models implies a change to the actual constituent systems



- Changed / new requirements become inputs to acquisition processes
 - Modify existing systems
 - Implement new systems

Control vs. Influence

- Traditional systems typically rely heavily on centralized command and control
 - Single acquisition authority
 - Prime contractor
 - Subcontractors via contractual arrangement
 - Suppliers
 - Other stakeholders
- SoSs rely on influence and indirect control
 - Multiple acquisition authorities
 - May be a SoS Integrator
 - Multiple System Contractors
 - Several additional stakeholders





Designing for **A**daptability and evolution**N** in
System of systems **E**ngineering

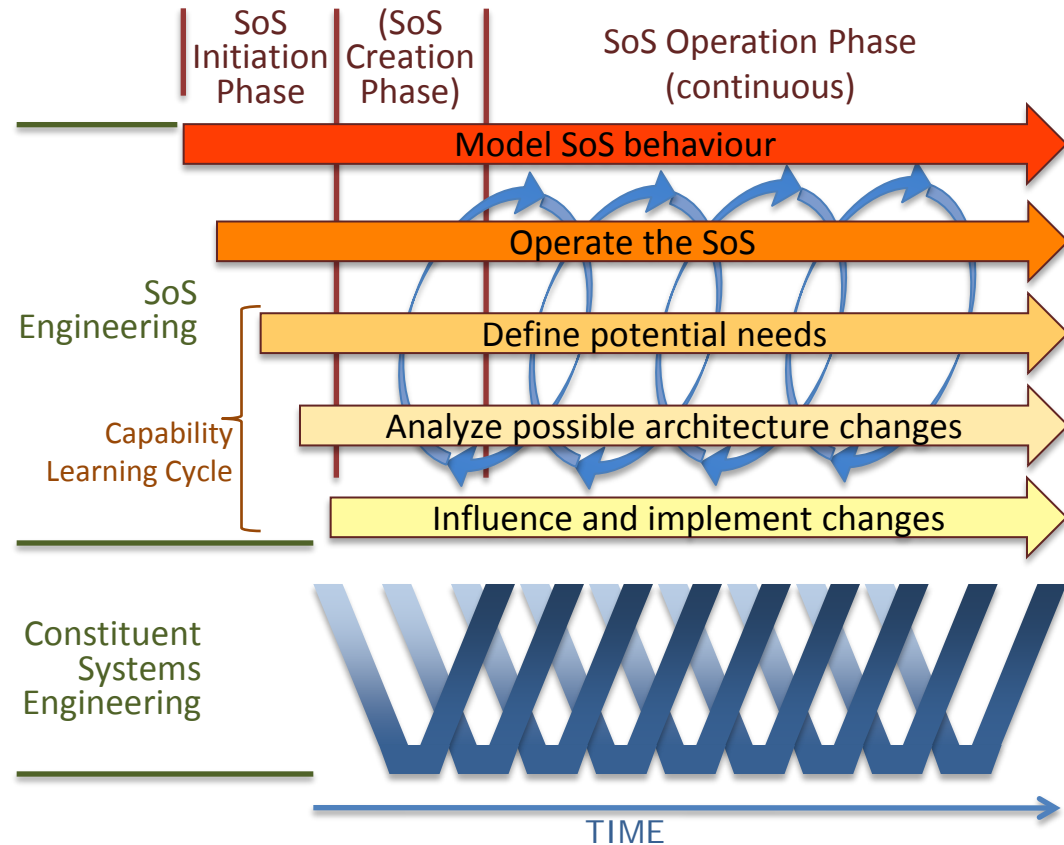
Summary

An effective methodology for SoS evolution
supported by useful tools

DANSE Methodology

Single model to embody the integrating thoughts

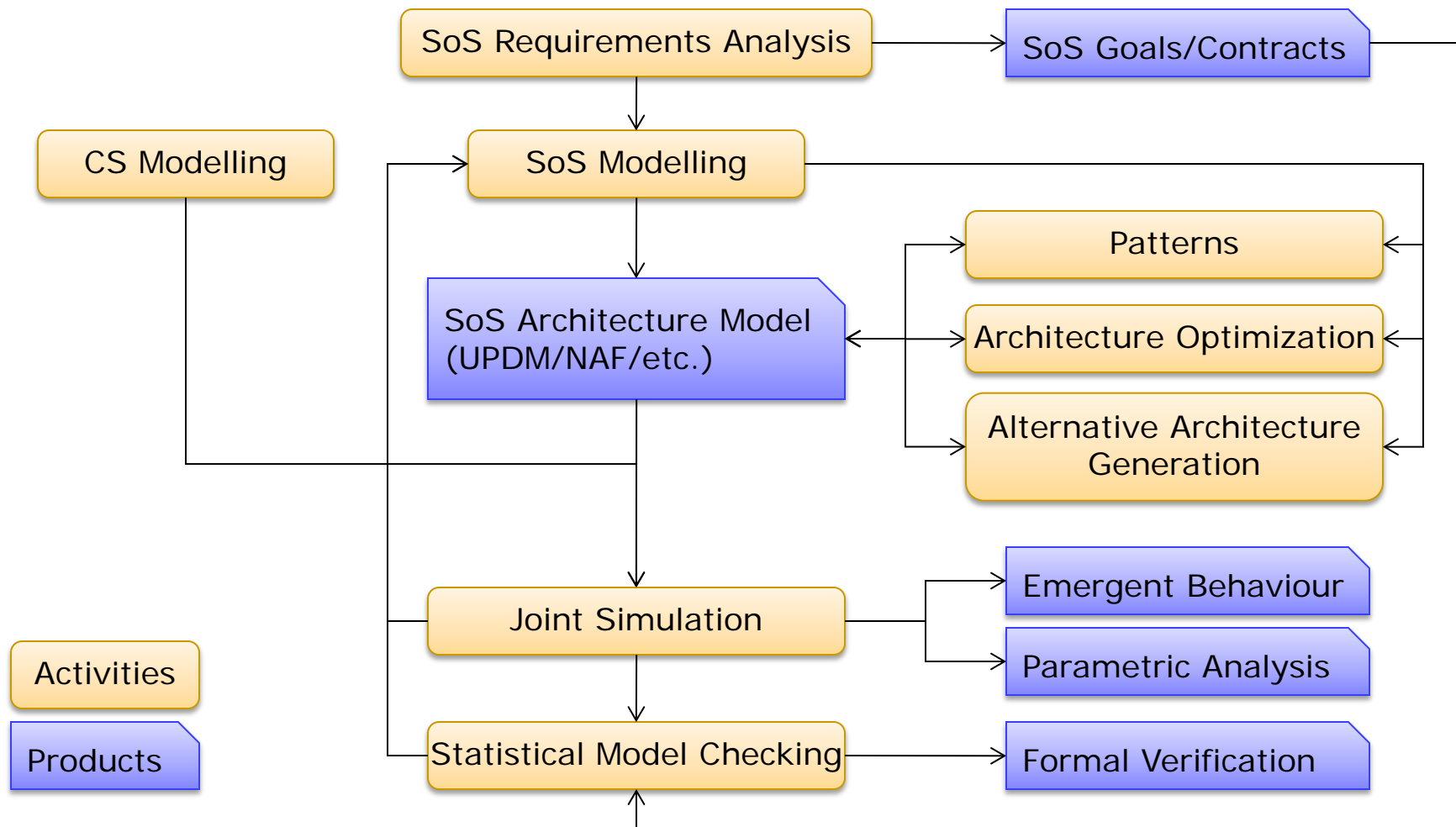
- An initiation phase
- Optional creation phase
- Forward movement through the SoS life
- Constant cycling of events/scenarios
- A “capability learning cycle”
 - Where the DANSE benefit happens!
- Normal Vee-based SE in the constituent systems



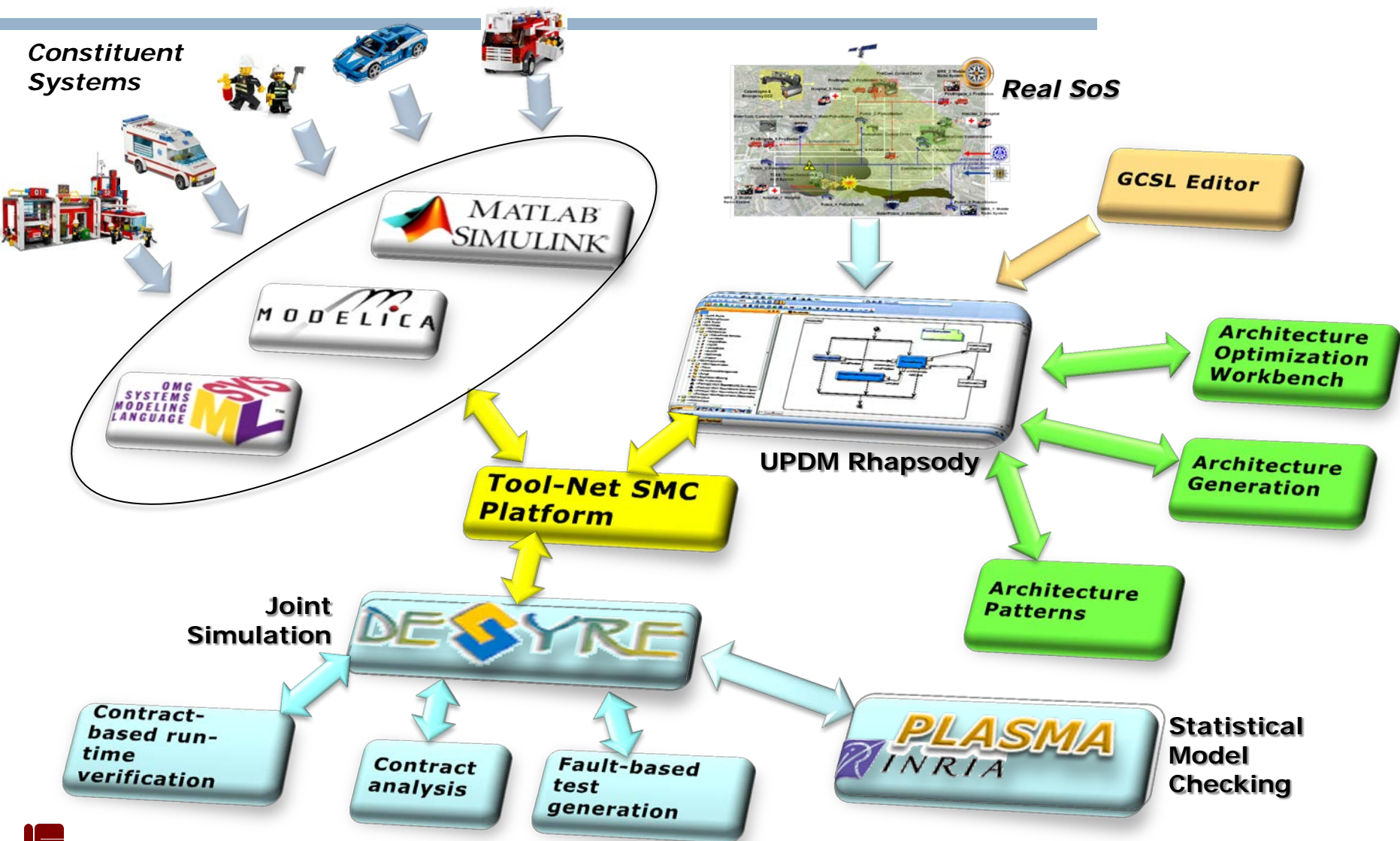
Alternate starting points:

- SoS is acknowledged among existing systems
- SoS is created by a Lead System Integrator

Example “Use Case” of Methodology



DANSE Tools



Designing for Adaptability and evolution in System of systems Engineering

