DANSE

Designing for **A**daptability and evolutio**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.

SEVENTH FRAMEWORK PROGRAMME

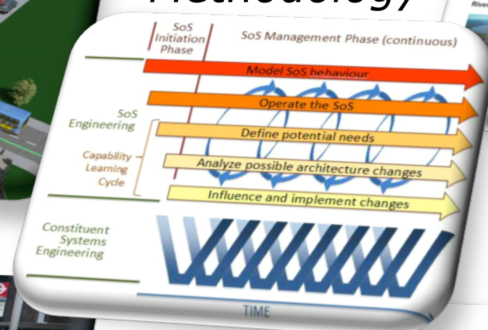
Eric Honour
+1 (615) 614-1109
ehonour@hcode.com

Agenda

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

Integrated Water Treatment and Supply

DANSE Methodology



Automated Ground Transport

Emergency Response

Air Traffic Management

DANSE

Designing for **A**daptability and evolutio**N** in **S**ystem 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?

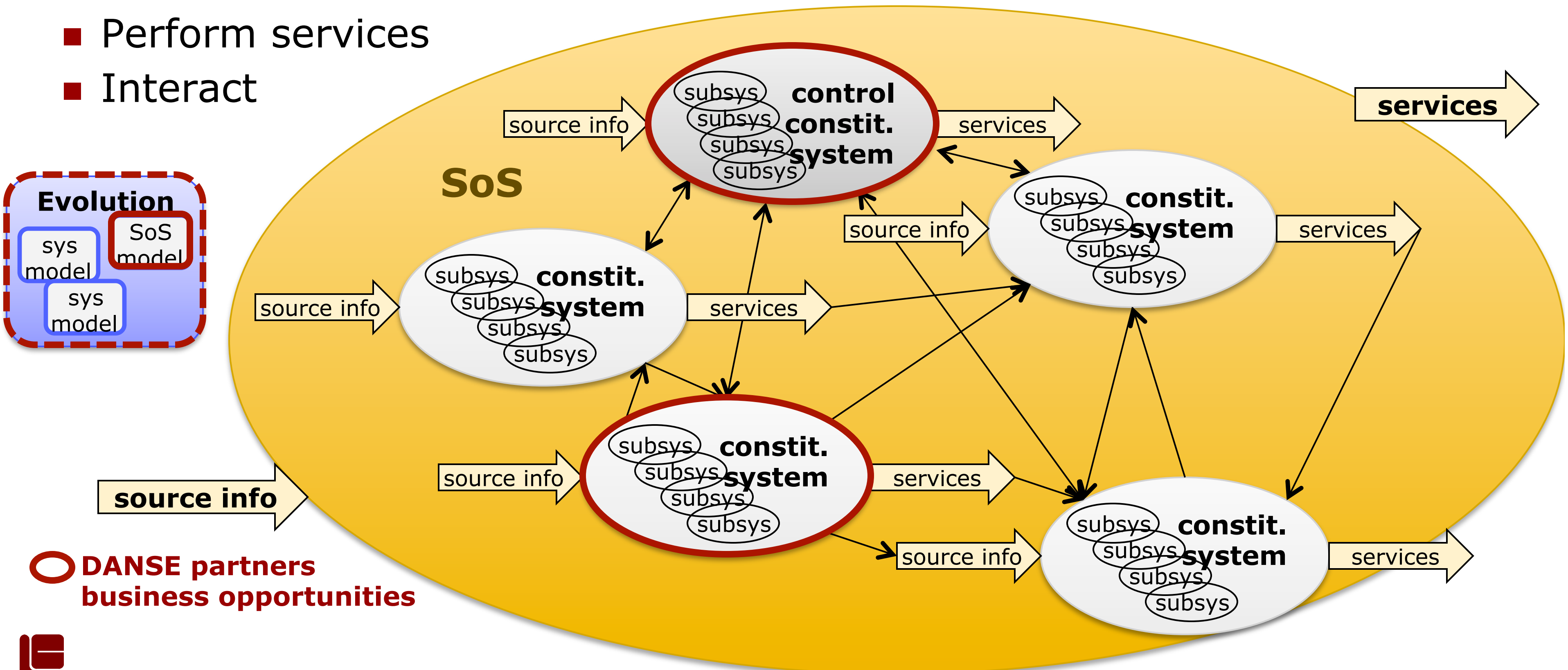
Architecture of an SoS

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



SoS Characteristics

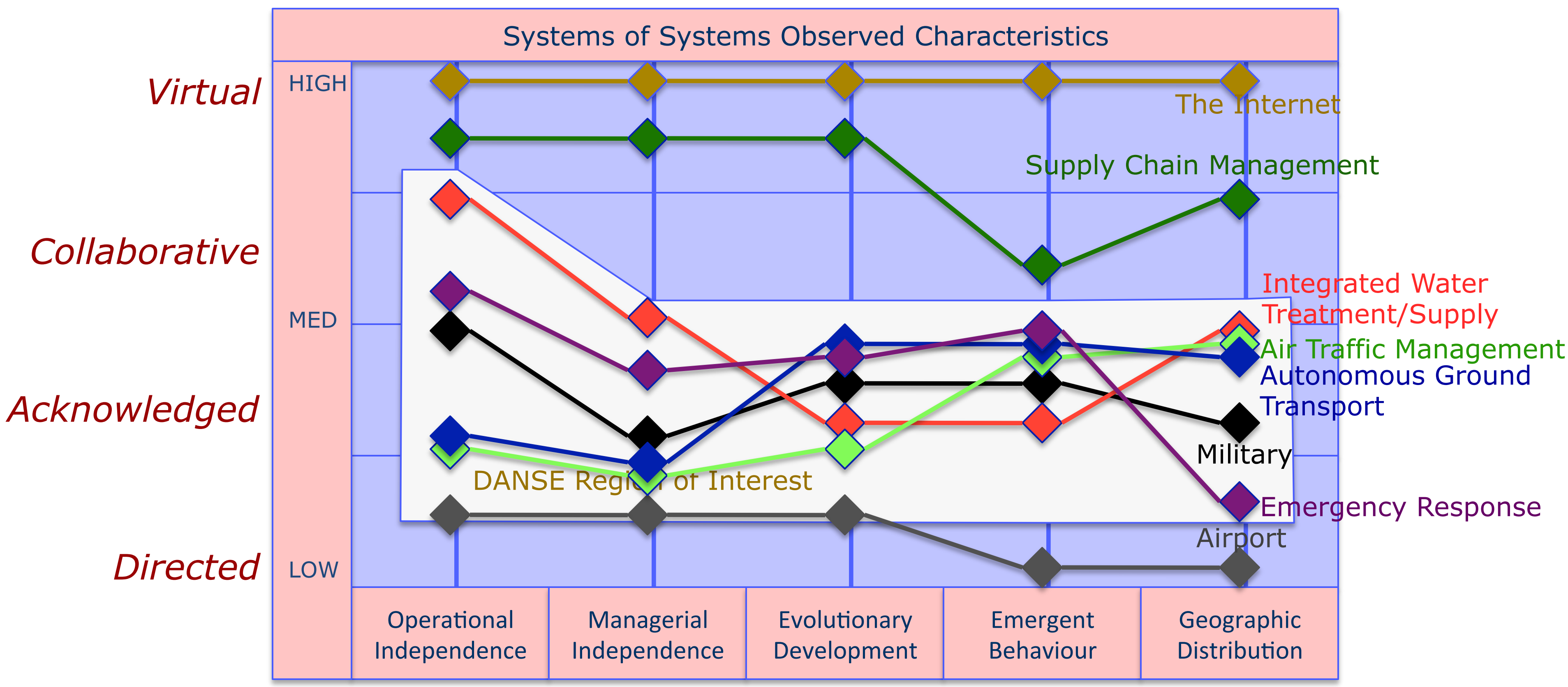
A System is a "System of Systems" if it exhibits significant amounts of:

- 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)

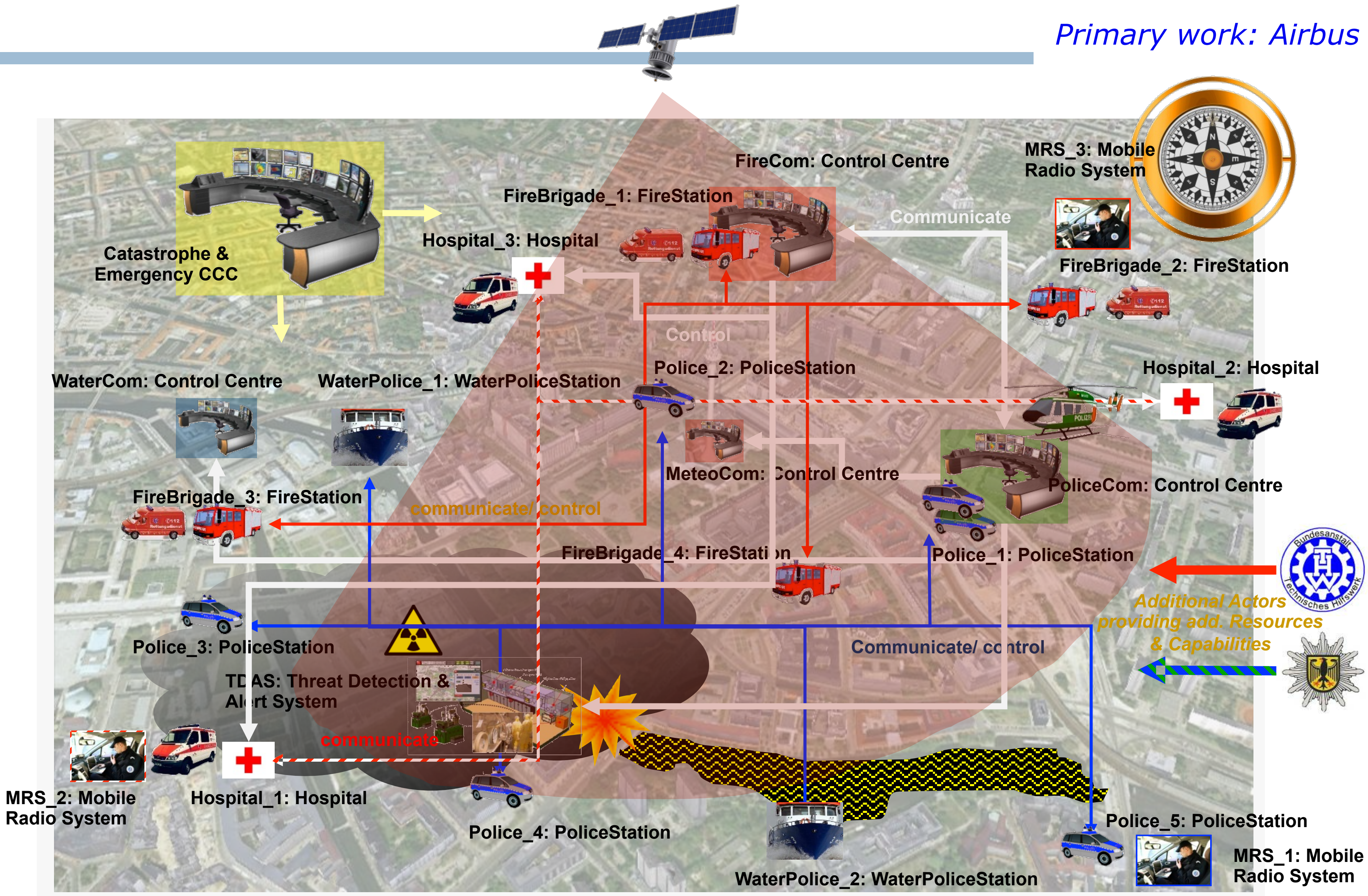
Differing Levels of "SoS-ness"

Primary work: Honourcode



Emergency Response SoS

Primary work: Airbus



Honourcode, Inc.

Emergency Response SoS Dynamicity

Primary work: Airbus

SoSE Challenges

Modelling the SoS | **Design Exploration** Architecture Alternatives | **Run Time Analysis & Simulation**

SoS operational timeline and dynamicity aspects

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

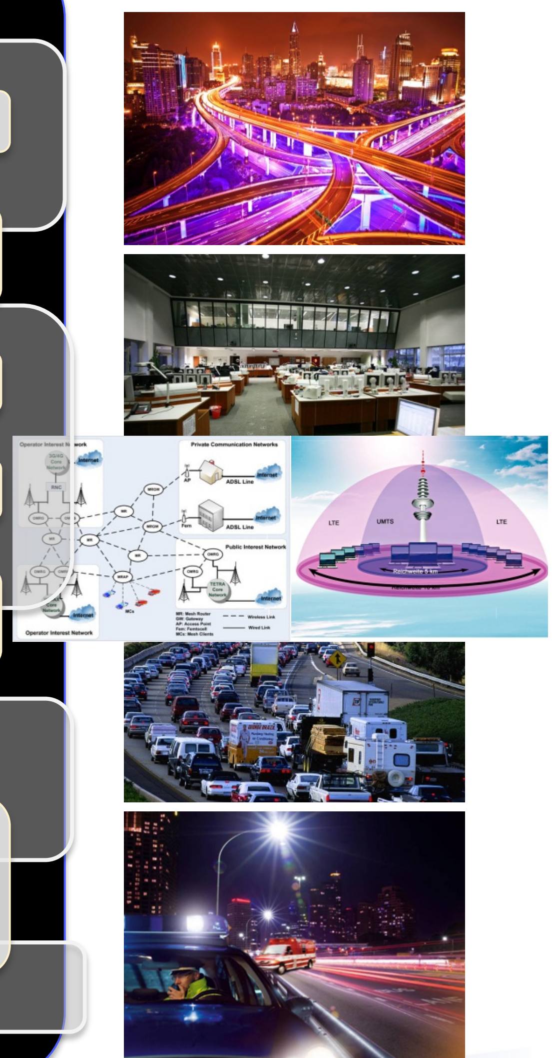
New C4I command & control organization & communication system (e.g. introduction of LTE)

Hours

Operational Dynamicity

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

Minutes



Designing for **A**daptability and evolutio**N** in **S**ystem of systems **E**ngineering

DANSE Methodology

What is the DANSE project?

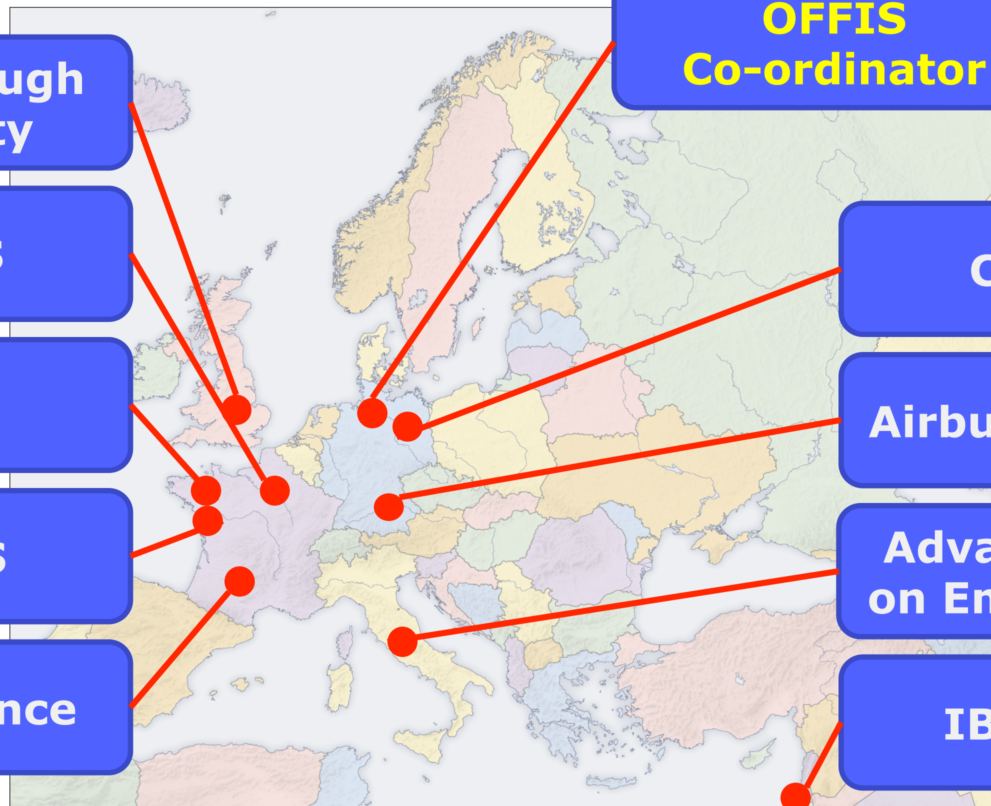
What is the life cycle of an SoS?

How does the DANSE methodology work in that life cycle?

DANSE Consortium

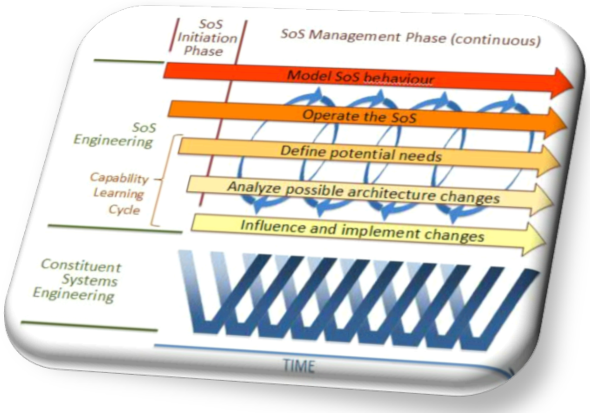
Contact:
Bernhard Josko
josko@offis.de

- Loughborough University
- THALES
- INRIA Rennes
- SODIUS
- Airbus France
- OFFIS Co-ordinator
- Carmeq
- Airbus Germany
- Advanced Laboratory on Embedded Systems
- IBM Haifa
- Israel Aerospace Industries
- Honourcode (technical support)



SEVENTH FRAMEWORK PROGRAMME

DANSE Project



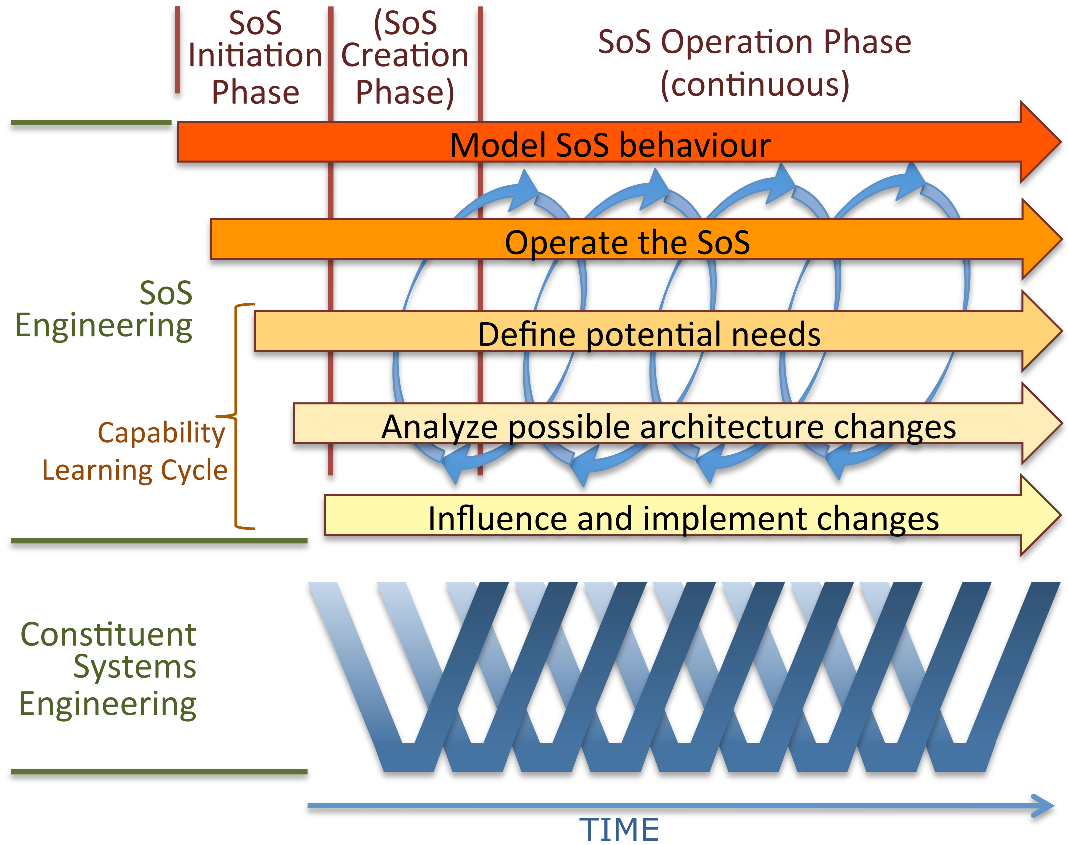
- Develop approaches for SoS engineering (design + manage)
  - *Methodology* to support evolution, adaptive and iterative SoS lifecycle
  - *Contracts* as semantically-sound model for SoS interoperations
  - *Architecting Approaches* for SoS – continuous and non-disruptive constituent system integration
  - *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 SoS Life Cycle

Primary work: Honourcode

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



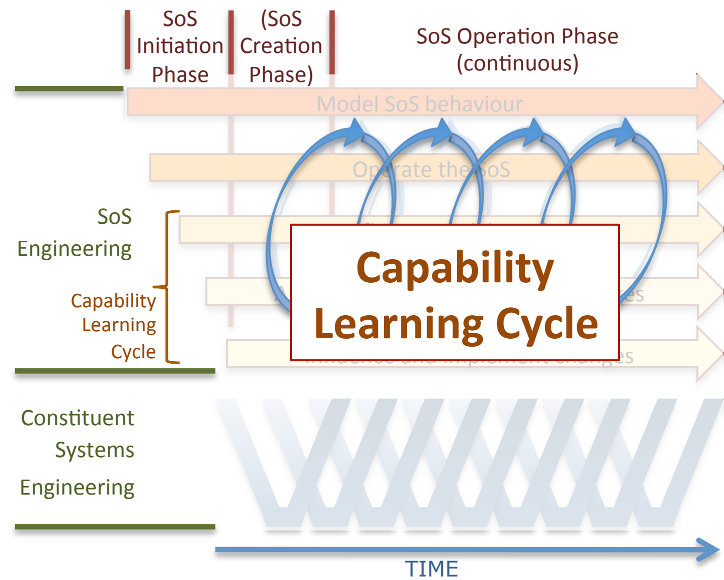
Alternate starting points:

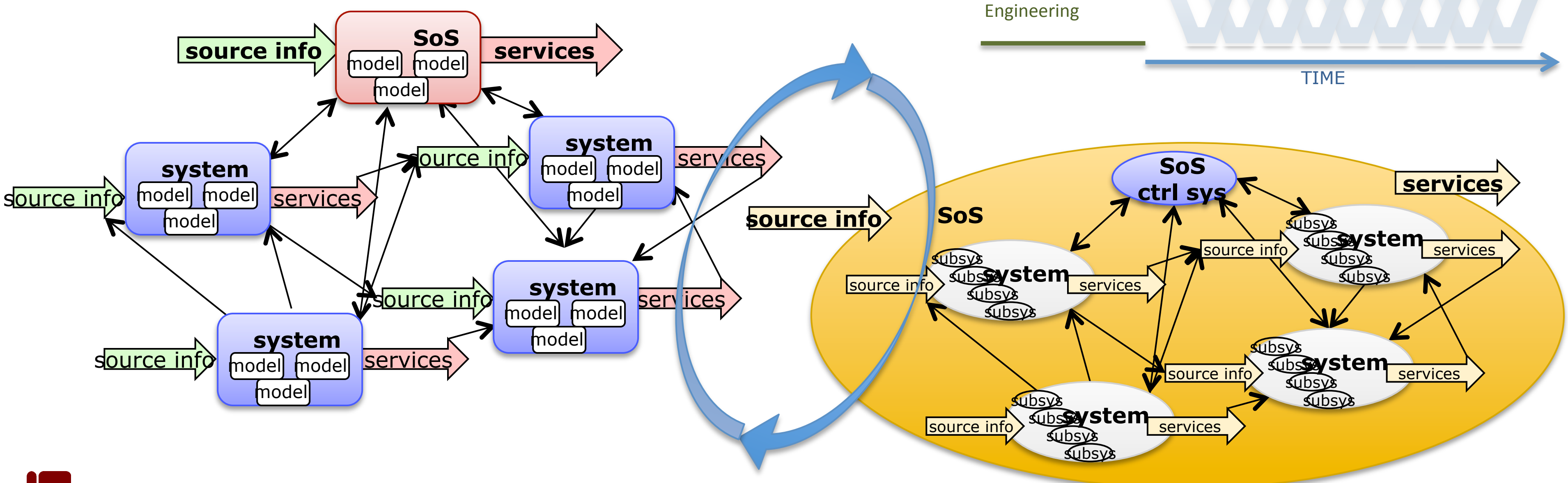
- SoS is acknowledged among existing systems
- SoS is created by a Lead System Integrator

Capability Learning Cycle

Primary work: Honourcode

- 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 evolutio**N** in **S**ystem 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

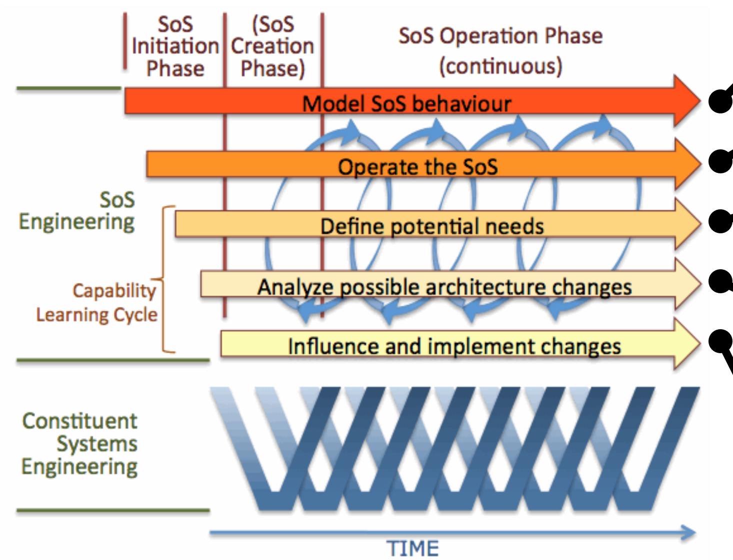
Solution Methods in the Lifecycle

Primary work: Honourcode

- 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

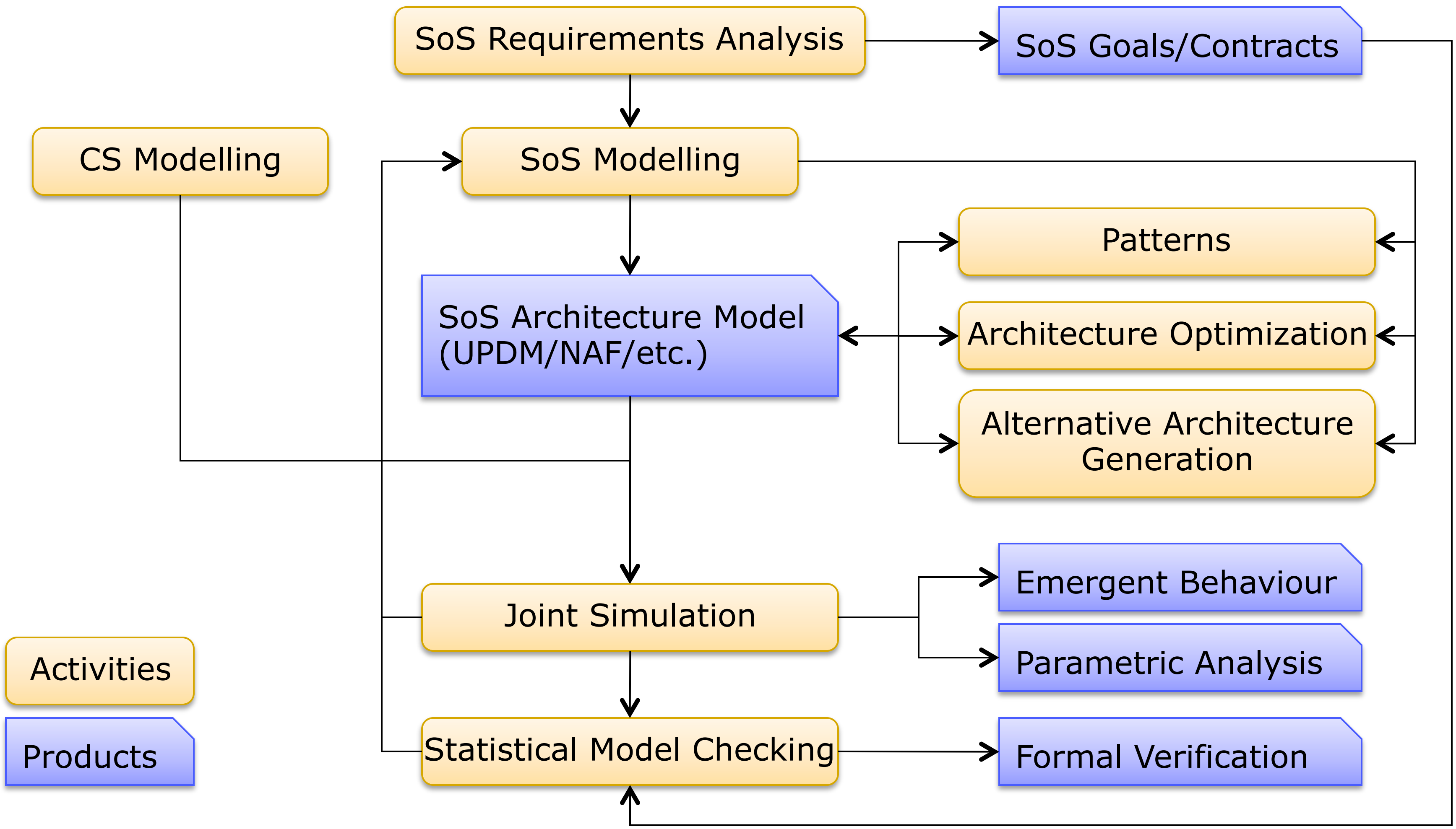


- Evaluate goals and contracts
- Apply architecture patterns
- Generate architecture alternatives
- Generate optimized architectures
- Optimize SoS architecture
- Perform joint simulation
- Evaluate emergent behaviour
- Perform statistical model checking
- Perform formal verification
- Share models

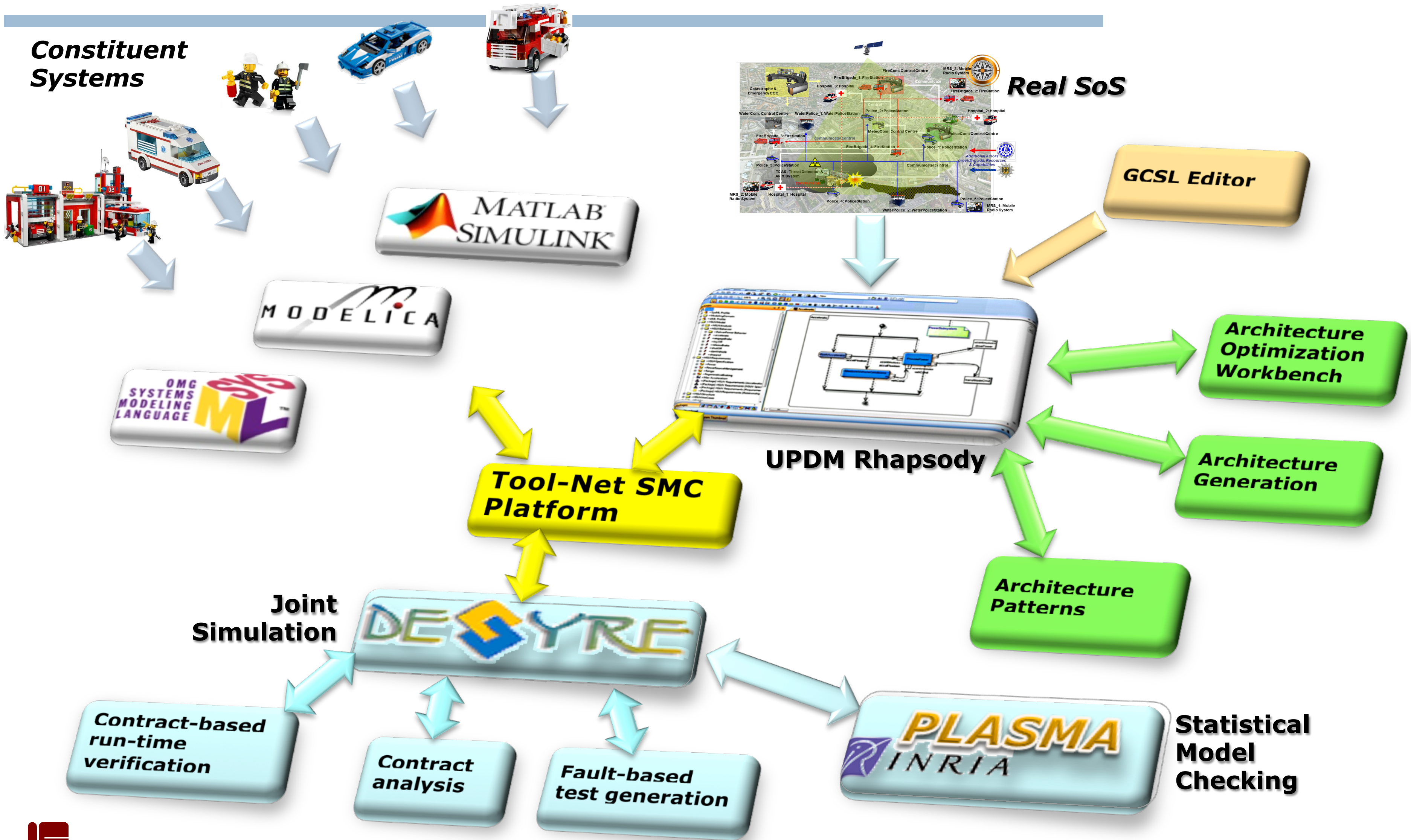
- Evaluate goals and contracts
- Optimize SoS architecture
- Perform joint simulation
- Evaluate emergent behaviour
- Perform formal verification
- Share models

Example "Use Case" of Methodology

Primary work: Honourcode



DANSE Tools


Constituent Systems
Real SoS
GCSL Editor
MATLAB SIMULINK
MODELICA
OMG SYSTEMS MODELING LANGUAGE
SysML
Architecture Optimization Workbench
UPDM Rhapsody
Architecture Generation
Tool-Net SMC Platform
Architecture Patterns
Joint Simulation
DESYRE
Contract-based run-time verification
Contract analysis
Fault-based test generation
PLASMA
INRIA
Statistical Model Checking

Designing for **A**daptability and evolutio**N** in **S**ystem 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

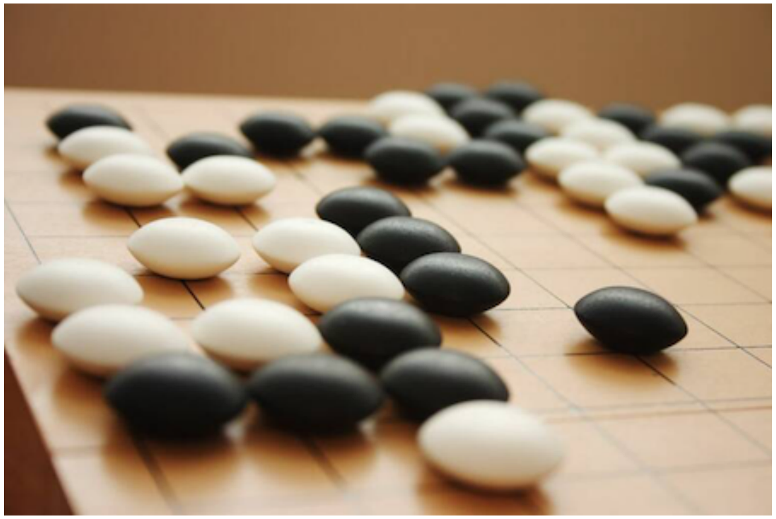
- SoS model should be executable as a simulation
  - Compare results with real world
  - Project "what if" scenarios
- These views support execution, lead to joint simulation

View	Name	Simulation
OV-5a	Operational Activity Decomposition Tree	SysML BDD, IBD: 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	SysML BDD, IBD: 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

Progressive Level of Detail

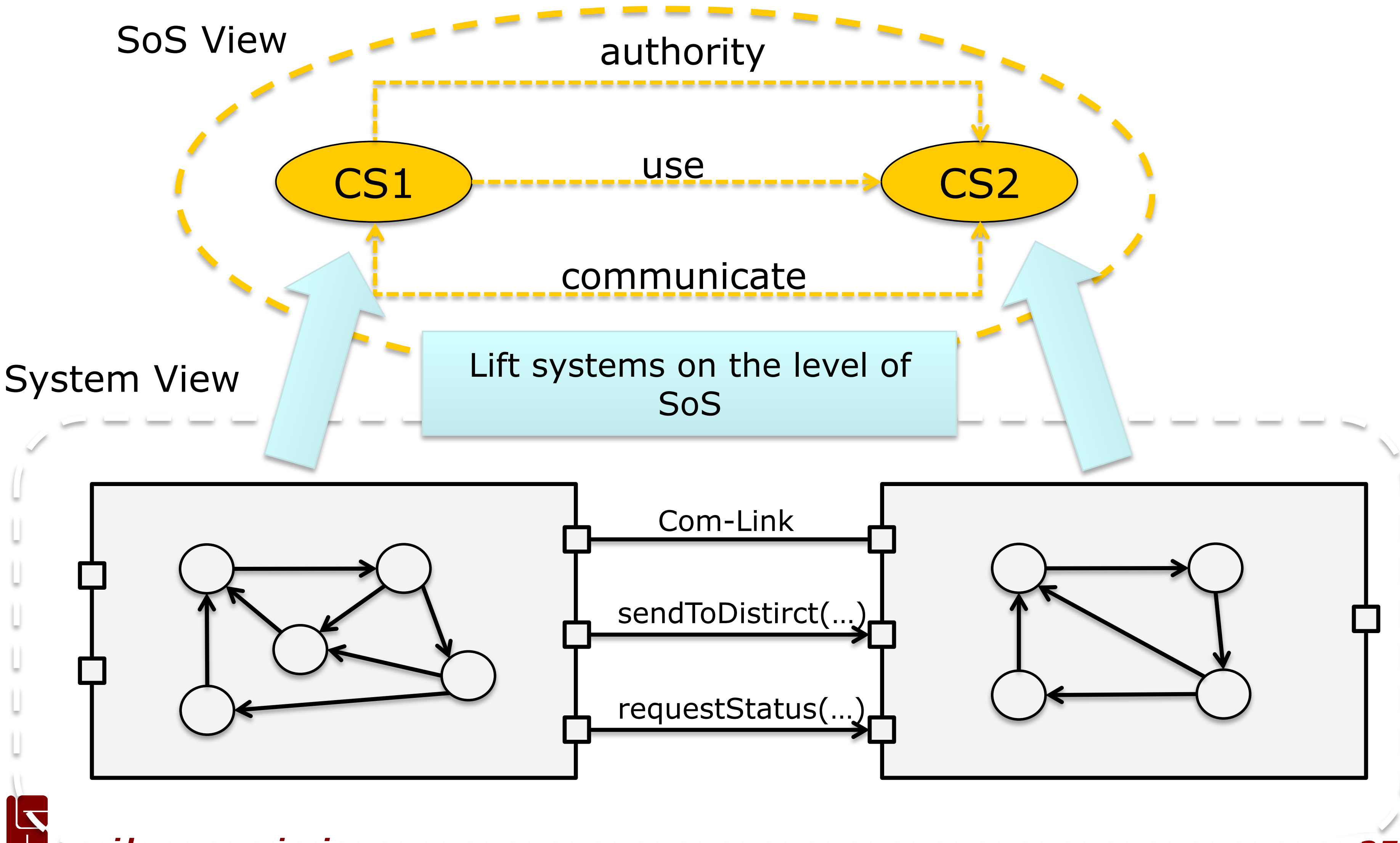
Primary work: Thales

- No "Big Bang" – complexity of SoS prevents ability to create a full SoS model
  - Constituent systems changing on their own
  - Do the math … SoS typically changes monthly!
- Create SoS models at progressive levels of detail
  - High level abstracted representation
    - Quick simulations, moderate accuracy, few emergent behaviours
  - States and modes representation
    - More detail using abstracted CS models
  - Details from CSs as available
    - Best simulation, richest emergent behaviours



Constituent System Models

Primary work: OFFIS



Abstraction Methods

Primary work: OFFIS

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"

Primary work: OFFIS

- 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

Primary work: OFFIS

- 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

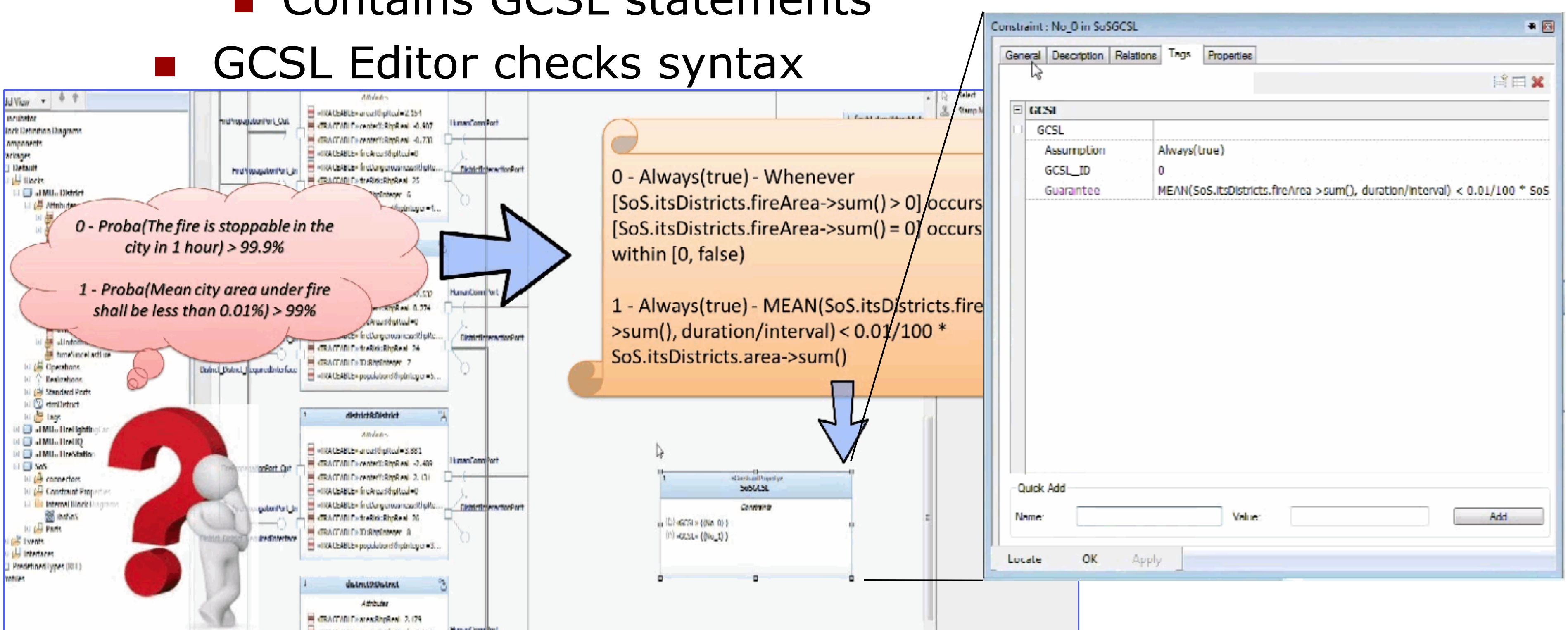
↓

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

GCSL Editor

Primary work: OFFIS

- 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



Solution Methods

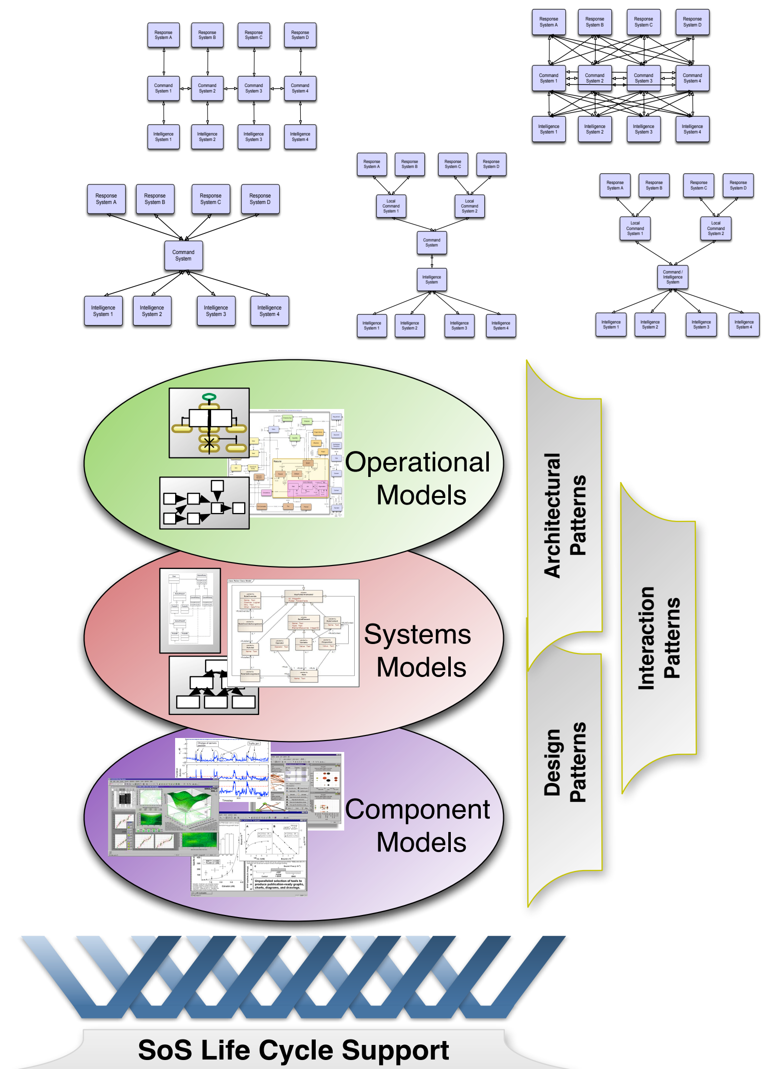
Architecture Exploration

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

Primary work: Loughborough Univ

- 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

Primary work: Loughborough Univ

Name of Pattern

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.

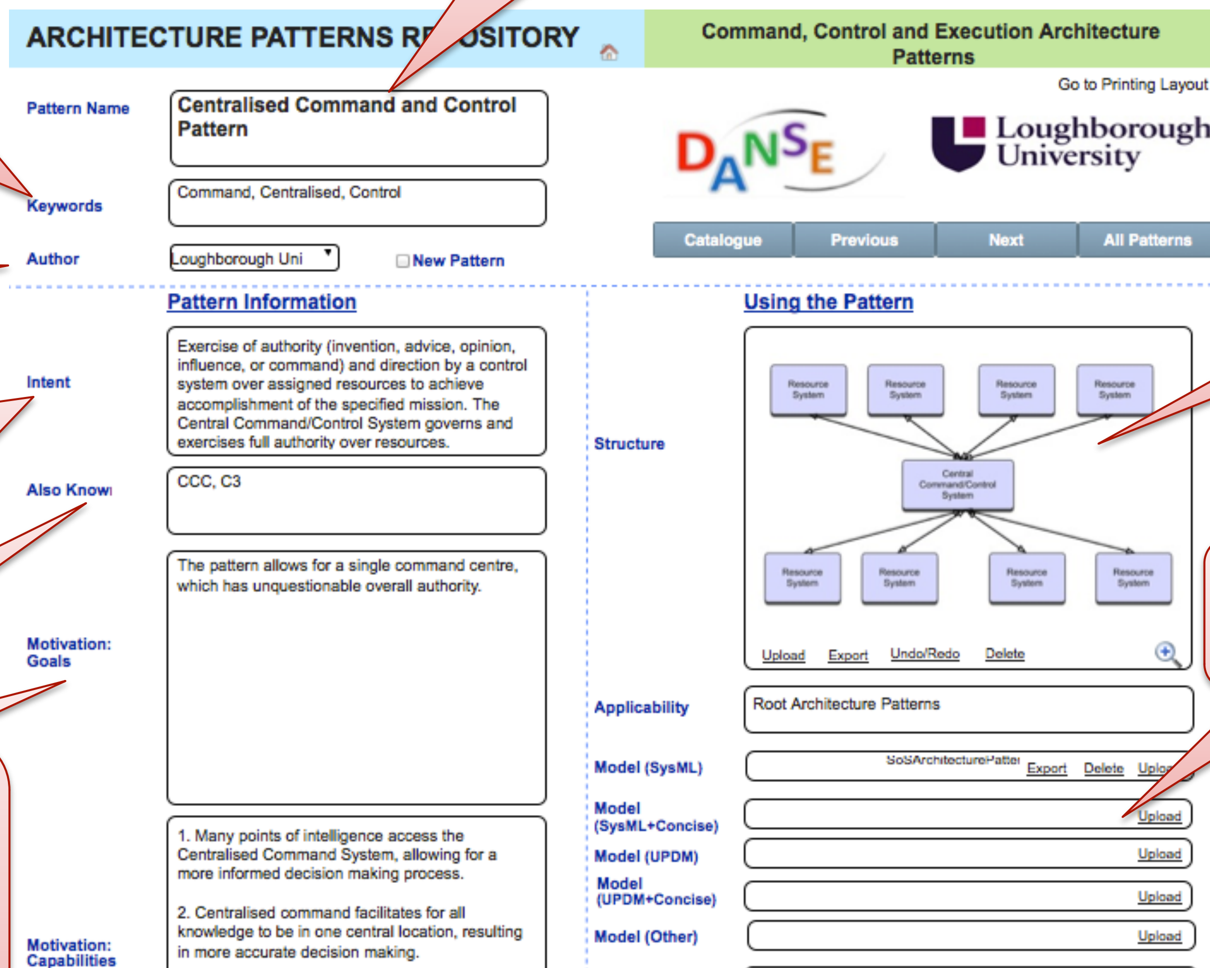


Diagram of Pattern's Structure

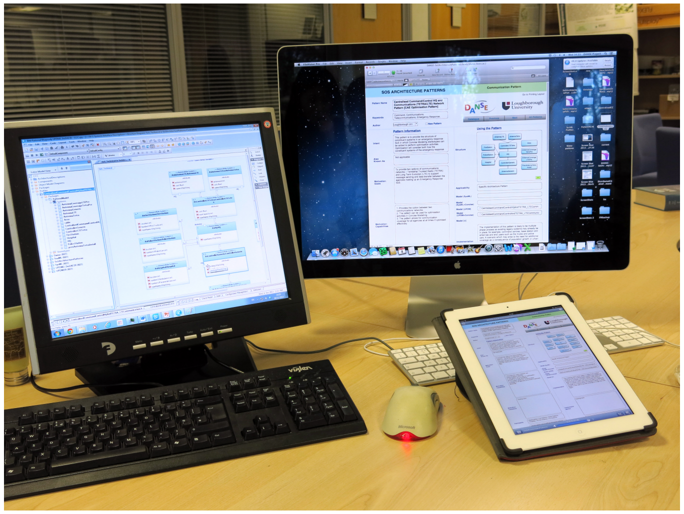
Rhapsody Models Available for Download

14 other fields also available

Architecture Patterns Repository

Primary work: Loughborough Univ

- 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.



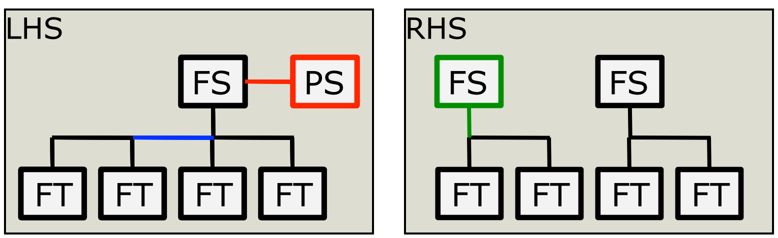
- Accessed via:
  - Conventional web browser (all popular browsers supported),
  - Apple iPad running the free FileMaker App – FileMaker Go.
  - User run-time version of FileMaker

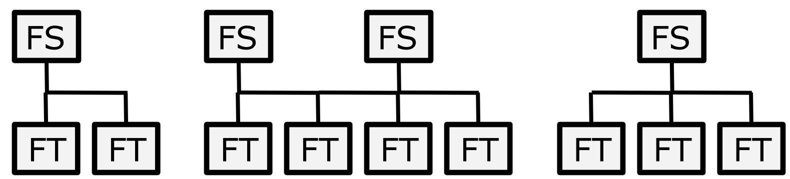
Graph Grammar

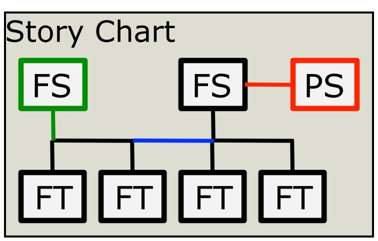
Primary work: OFFIS

- 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*

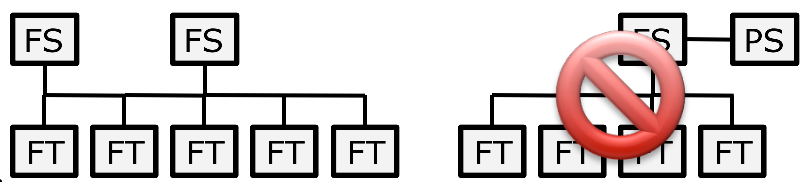
1. **Reader**: Matched, not changed.
2. **Eraser**: Matched and **removed**.
3. **Creator**: **Added** to the model.
4. **Embargo**: **Prevents the match**.







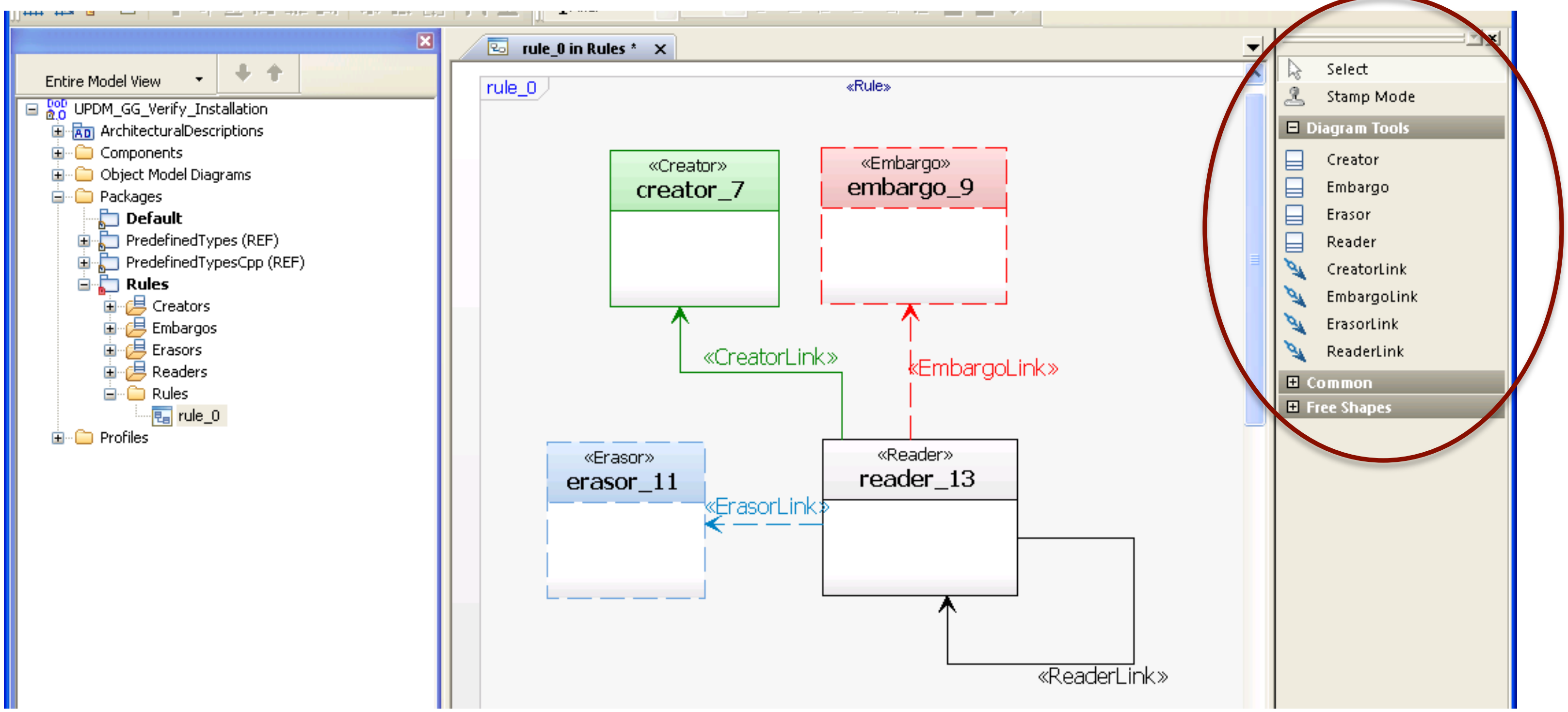
FT: Fire truck
FS: Fire station
PS: Police station



DANSE Graph Grammar

Primary work: OFFIS

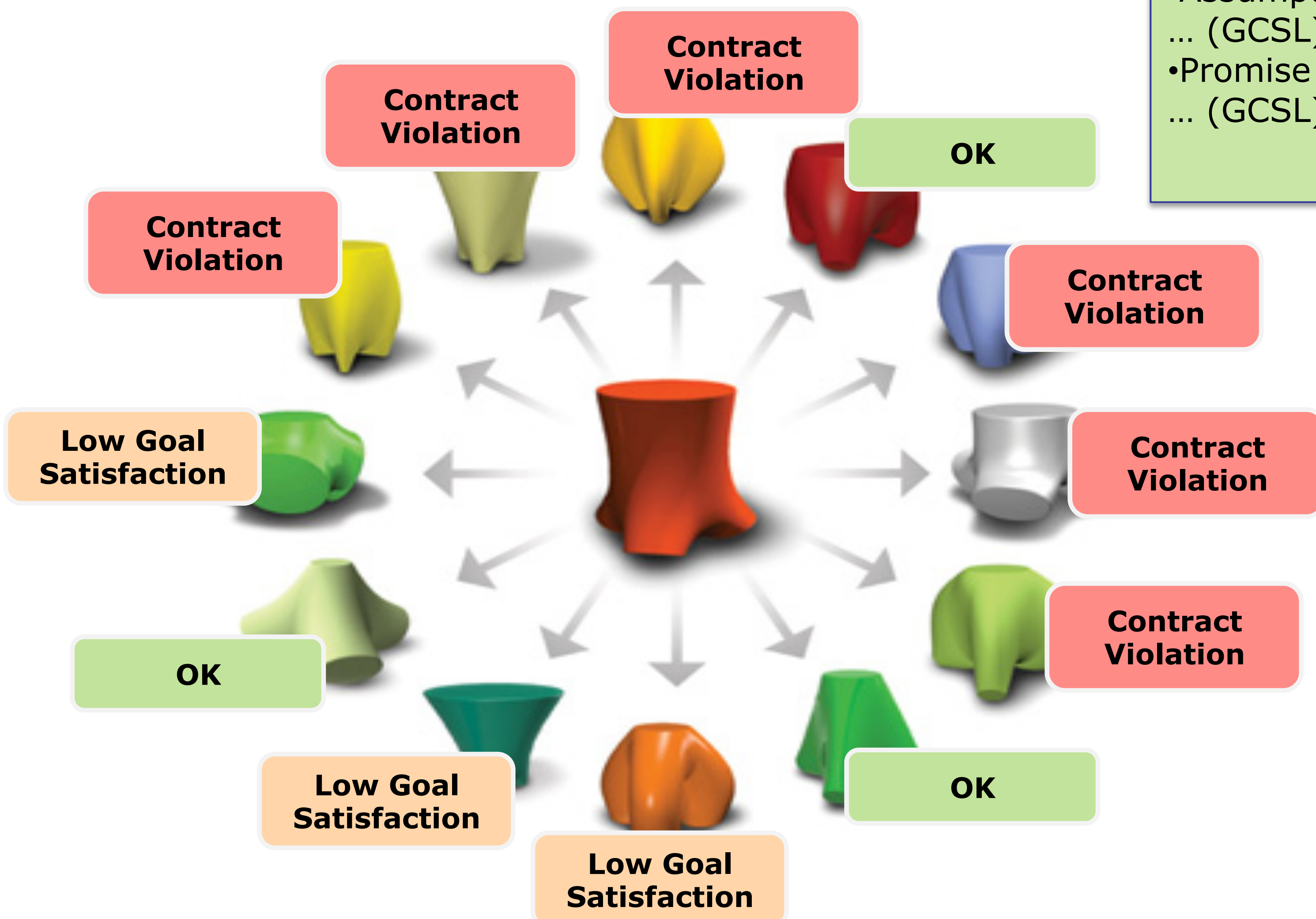
- Story Charts **implemented** as special UPDM diagrams
- Based on a UPDM **profile** to enable the modeling of a rule
- Revised models created automatically by applying the rules
  - Can match models in any BDD or IBD



Exploration of Design Space

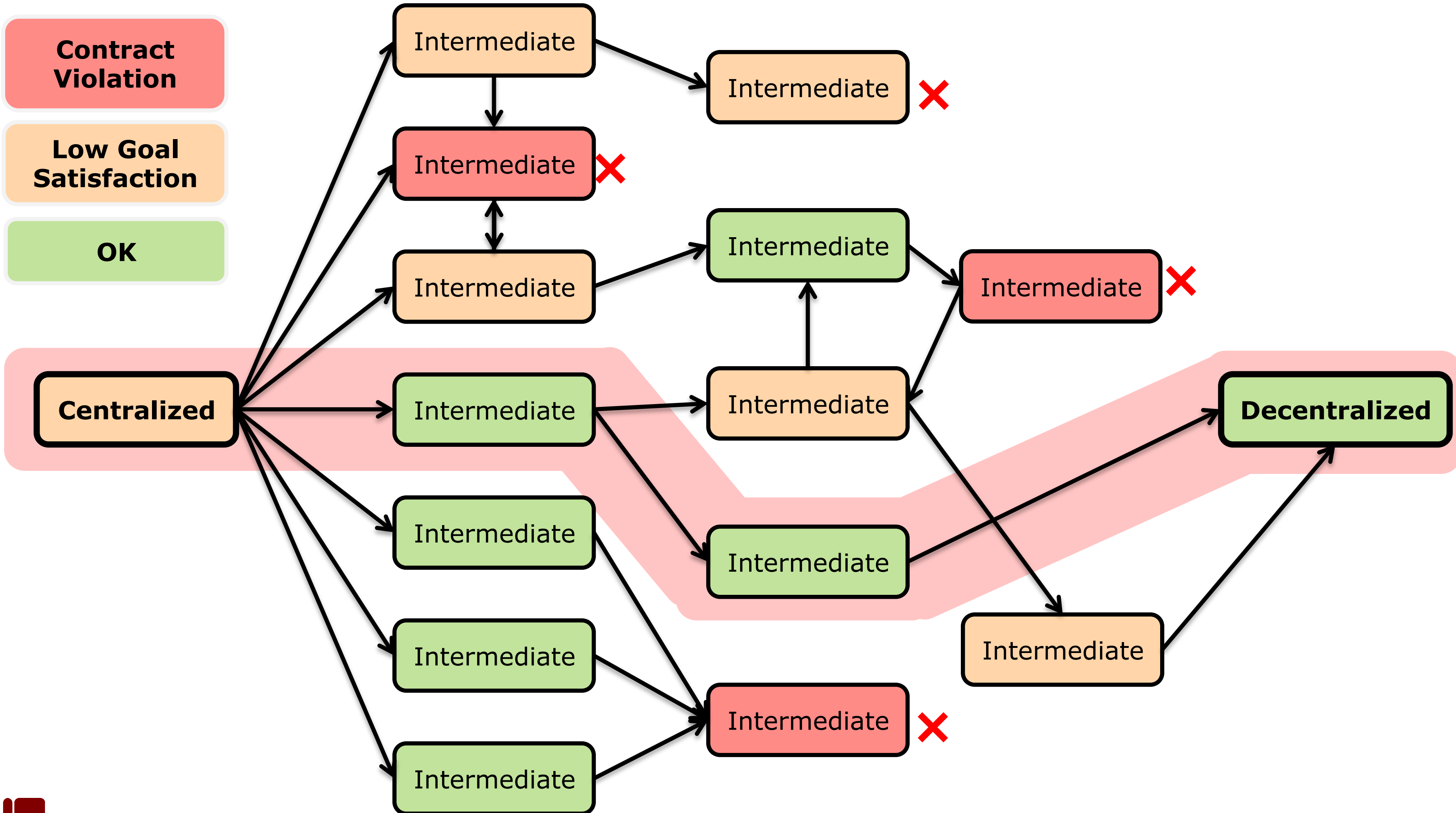
Primary work: OFFIS

SoS contract
- Assumption
… (GCSL)
- Promise
… (GCSL)



Reachability of Future Architectures

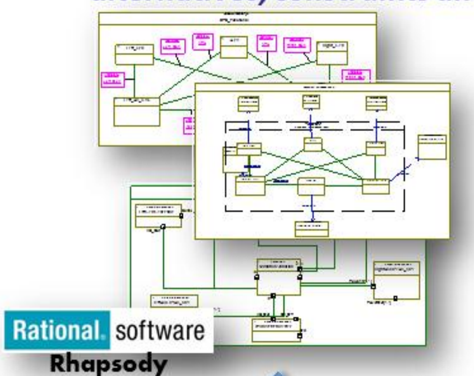
Primary work: OFFIS



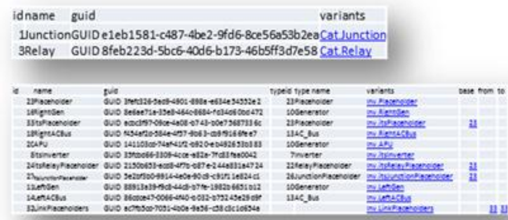
Architecture Optimization Concept

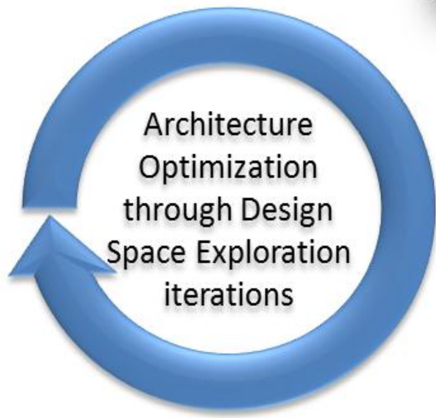
Primary work: IBM

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

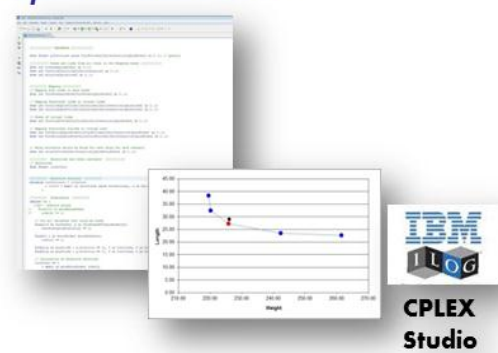


2. Derived Data Schema for Input and output structures

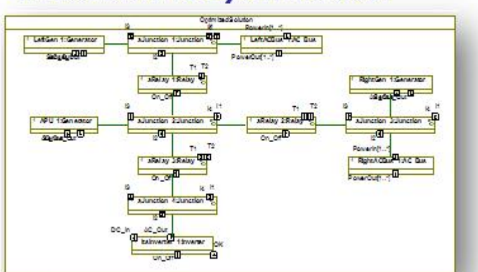




3. Automatic translation(via an interchange format) into Optimization solver

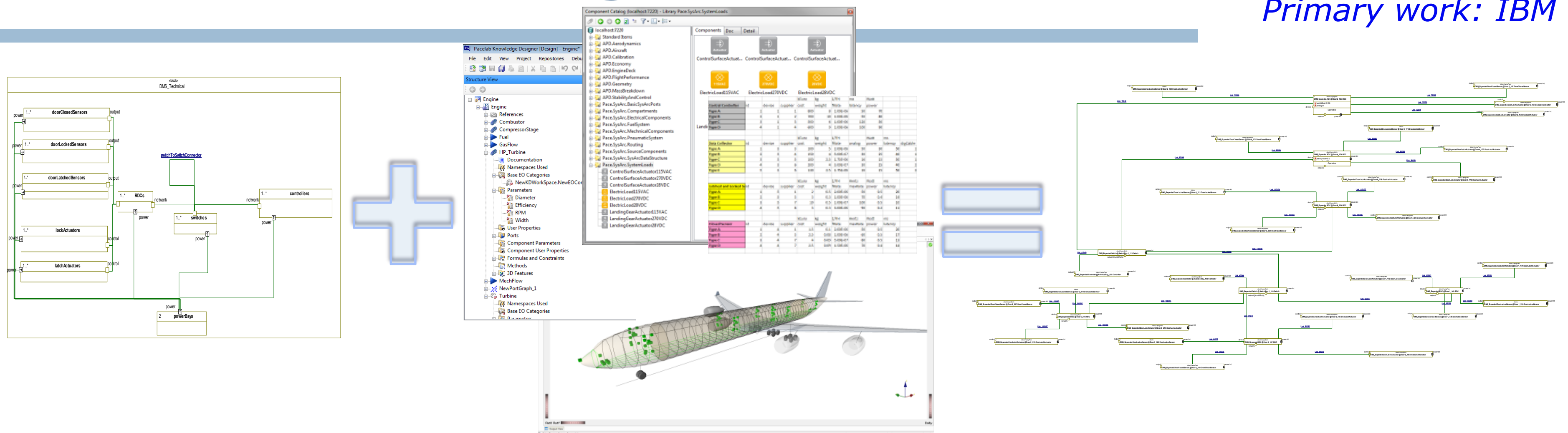


4. Optimized architecture back annotated to SysML model



Concise Modeling

Primary work: IBM



SysML models combined with tabular data

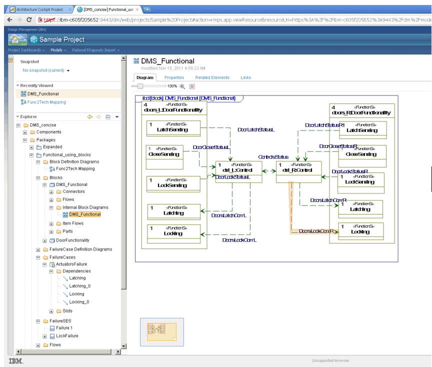
- SysML depicts the system composition rules (architectural template or pattern)
- Tables contain instantiations, variations in quantities or parameters

- Automatic Generation tool creates architecture variants by applying the table data to the template

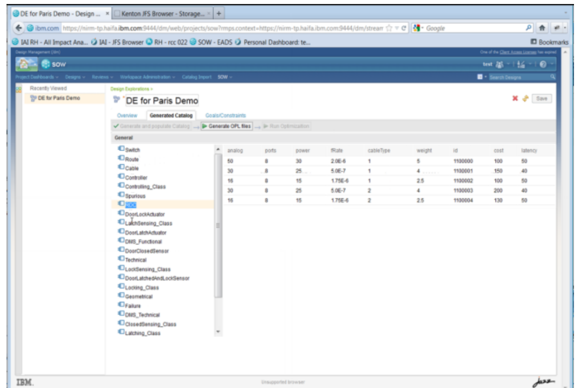
Dashboard for Architecture Optimization

Primary work: IBM

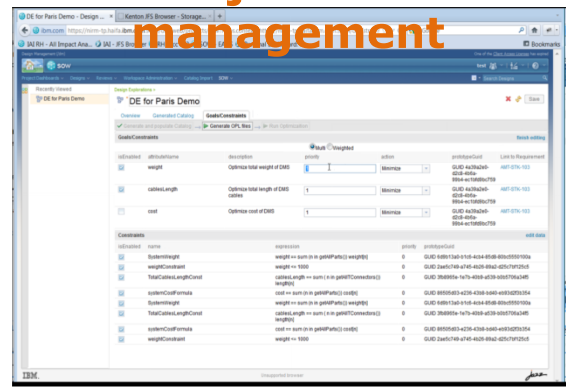
Modeling



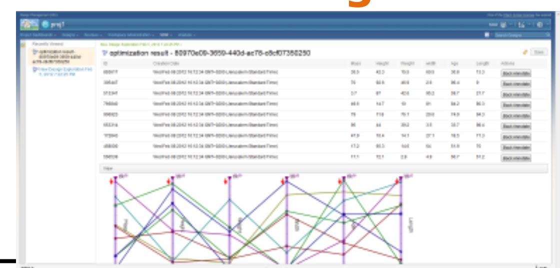
Data management



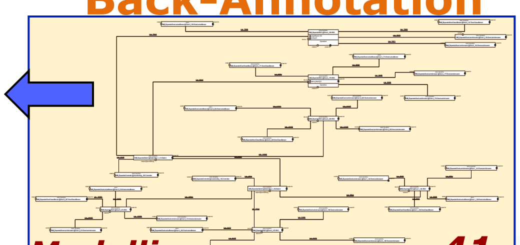
Objectives management



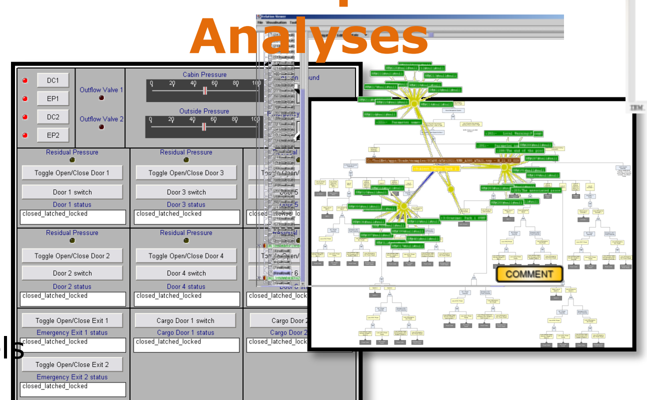
Results management



Back-Annotation



Multiple Analyses



- 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

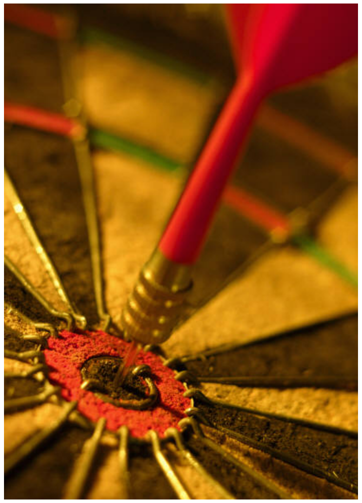
Solution Methods

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

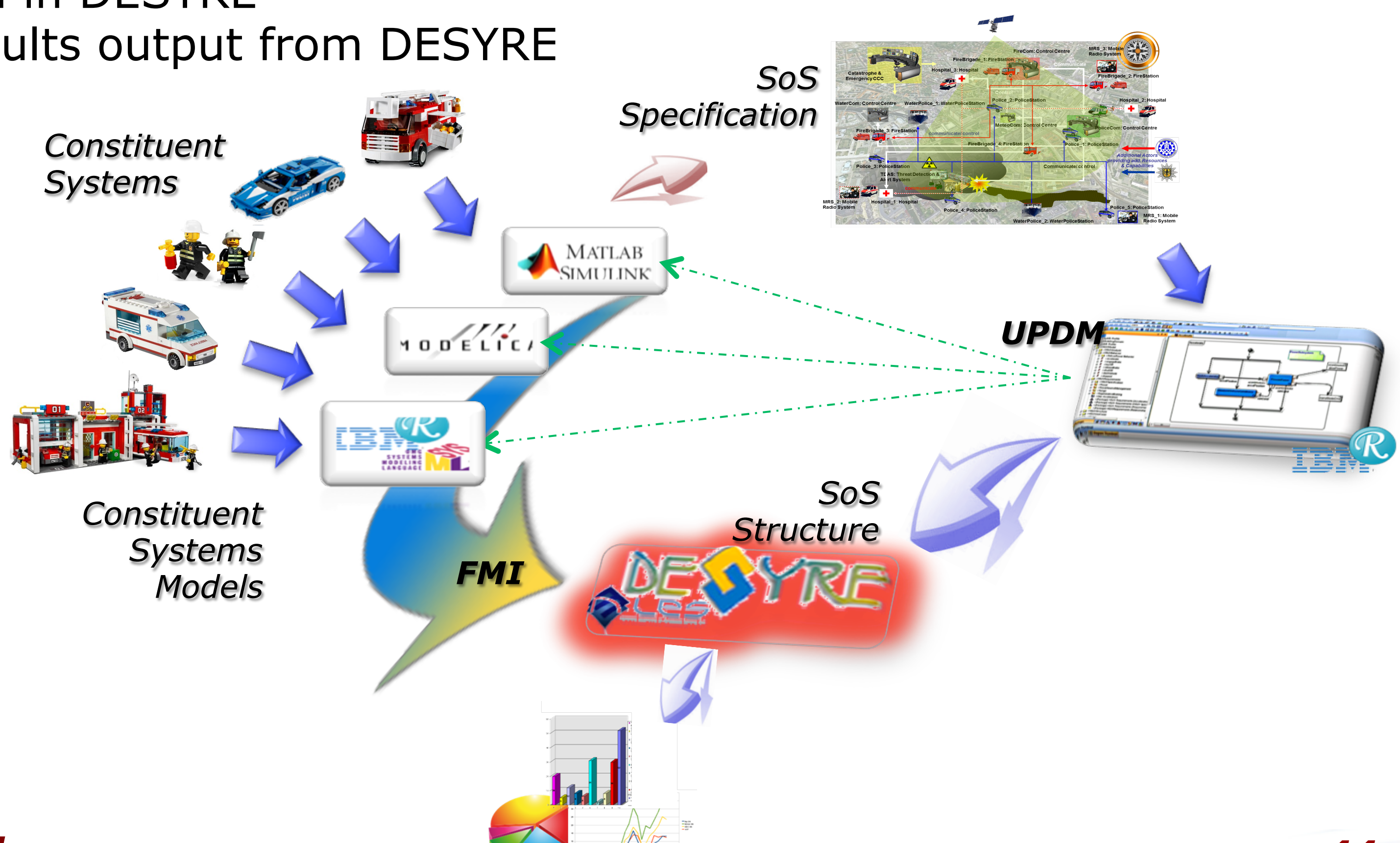
- Have generated multiple architecture alternatives
  - Patterns application
  - Graph grammar automated generation
  - Concise modeling with optimization
- Need SoS joint simulation to evaluate performance
  - Predict characteristics of interest
  - Evaluate contracts and goals during simulation
  -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



Joint simulation

Primary work: ALES

- 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

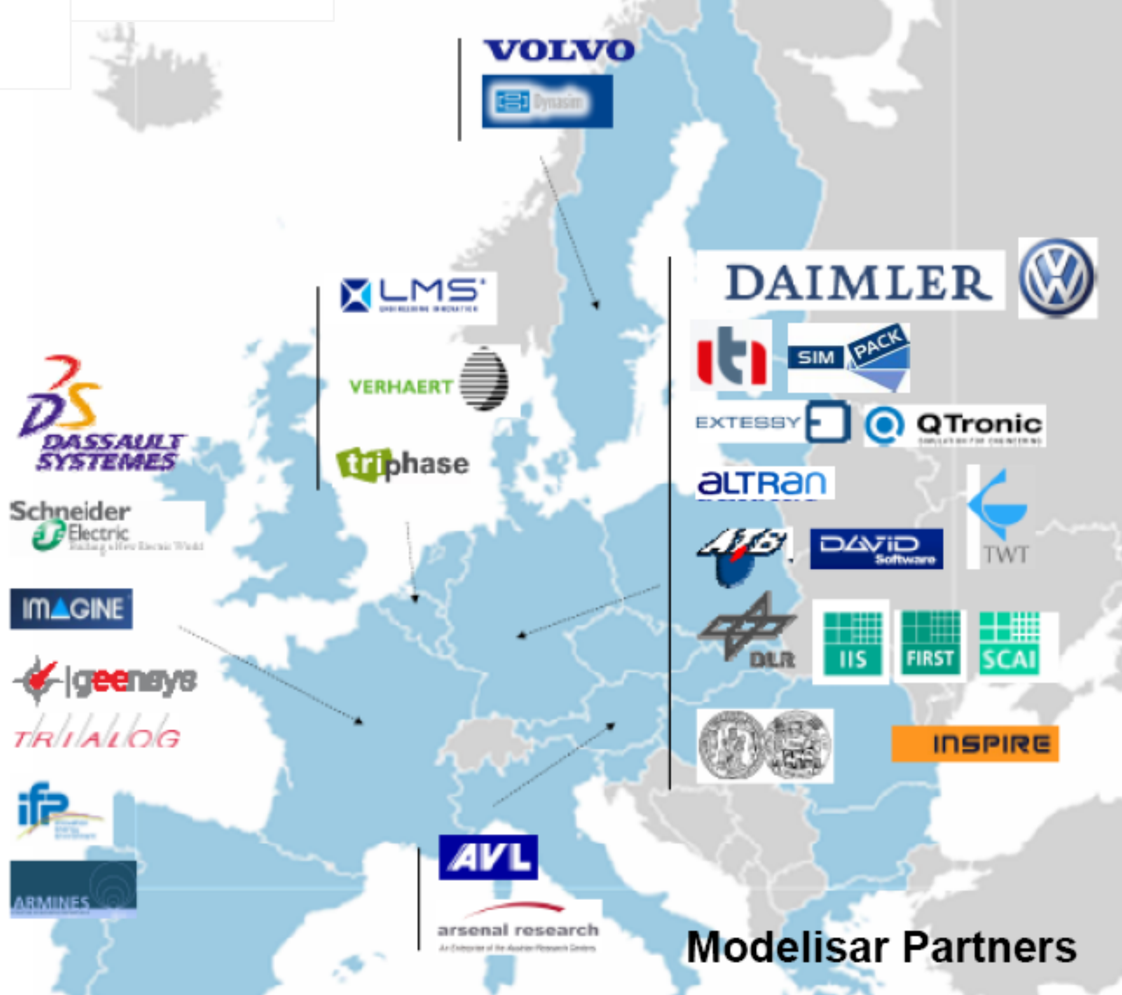


FMI Standard

Tools supporting FMI	Model-Exchange		Co-Simulation		Notes
	Export	Import	Slave	Master	
AMESim	Available	Available	Planned	Planned	Modelica environment from LMS-Imagine
ASIM	Planned		Planned		AUTOSAR Builder from Dassault Systèmes
Adams		Planned	Planned	Planned	High end multibody dynamics simulation software from MSC Software
Atego Ace		Available		Available	Co-simulation environment with AUTOSAR and HIL support
Building Controls Virtual Test Bed				Planned	Software environment, based on Ptolemy II, for co-simulation of, and data exchange with, building energy and control systems.
CATIA V6R2013	Available	Available	Available	Available	Environment for Product Design and Innovation, including systems engineering tools based on Modelica, by Dassault Systèmes
ControlBuild	Available	Available	Available	Available	Environment for IEC 61131-3 control applications from Dassault Systèmes
CosiMate		Available		Available	Co-simulation Environment from ChiasTek
Cybernetica CENIT		Available		Planned	Industrial product for nonlinear Model Predictive Control (NMPC) from Cybernetica.
Cybernetica ModelFit		Available		Available	Software for model verification, state and parameter estimation, using logged process data. By Cybernetica.
DSHplus	Planned		Planned		Fluid power simulation software from FLUIDON
Dymola 2013	Available	Available	Available	Planned	Modelica environment from Dassault Systèmes
EnergyPlus			Planned	Available	Whole building energy simulation program
FMI Library (from Modelon)		Available		Available	Open source (BSD) C library for integration of FMI technology in custom applications by Modelon.
FMI add-on for NI VeriStand	Available	Available		Available	Manages simulations with FMI for co-simulation V1.0, available from DOFware
FMU SDK	Available	Available	Available	Available	FMU Software Development Kit from QTronic.
FMU Trust Centre			Available		Cryptographic protection and signature of models including their safe PLM storage; secure authentication and authorization for protected (co-)simulation
IPG CarMaker (via FMI Toolbox for CARMAKER)		Planned		Planned	via FMI Toolbox for CARMAKER from Modelon (planned for the end of 2012).
JFMI			Available	Available	A Java Wrapper for the Functional Mock-up Interface, based on FMU SDK
JModelica.org	Available	Available	Planned	Planned	Open source Modelica environment from Modelon
MATLAB (via FMI Toolbox for MATLAB)		Available		Available	via FMI Toolbox for MATLAB from Modelon
MWorks 2.5	Available	Planned	Planned	Planned	Modelica environment from Suzhou Tongyuan
MapleSim	Available	Planned	Planned	Planned	Modelica-based modeling and simulation tool from Maplesoft
Microsoft Excel (via FMI Add-in for Excel)				Available	via FMI Add-in for Microsoft Excel by Modelon. Offers support for batch simulation of FMUs.
NI LabVIEW		Planned			Graphical programming environment for measurement, test, and control systems from National Instruments
NI VeriStand				Planned	Real-Time Testing and Simulation Software from National Instruments
OPTIMICA Studio	Available	Planned	Planned	Planned	Modelica environment from Modelon
OpenModelica	Available	Available			Open source Modelica environment from OSMC
Ptolemy II				Planned	Software environment for design and analysis of heterogeneous systems.
PyFMI		Available			For Python via the open source package PyFMI from Modelon. Also available as part of the JModelica.org platform.
SIMPACK 9	Planned	Available	Planned	Available	High end multi-body simulation software from SIMPACK AG
Silver 2.4.4		Available	Planned	Available	Virtual integration platform for Software in the Loop from QTronic
SimulationX 3.4	Available	Available	Available	Available	Modelica environment from ITI
Simulink (via @Source)	Available				via @Source
Simulink (via Dymola)	Available				via Dymola 2013 using Real-Time Workshop
Simulink (via FMI Blockset)				Available	Import of FMI Co-Simulation models into Simulink - provided by Claytex.
Simulink (via FMI Toolbox)		Available		Available	via FMI Toolbox from Modelon
TISC		Available		Available	Co-simulation environment from TLK-Thermo
TWT Co-Simulation Framework			Available	Available	Communication layer tool to flexibly plug together models for performing a co-simulation; front-end for set-up, monitoring and post-processing included
Vertex	Planned				Modelica environment from deltatheta
Virtual.Lab Motion	Planned	Available	Available	Available	Virtual.Lab Motion is a high end multi body software from LMS International
xMOD		Available		Available	Heterogeneous model integration environment & virtual instrumentation and experimentation laboratory from IFPEN distributed by D2T.

Over 35 tools support FMI

The FMI specification is developed within the ITEA-2 project Modelisar 2008 - 2011



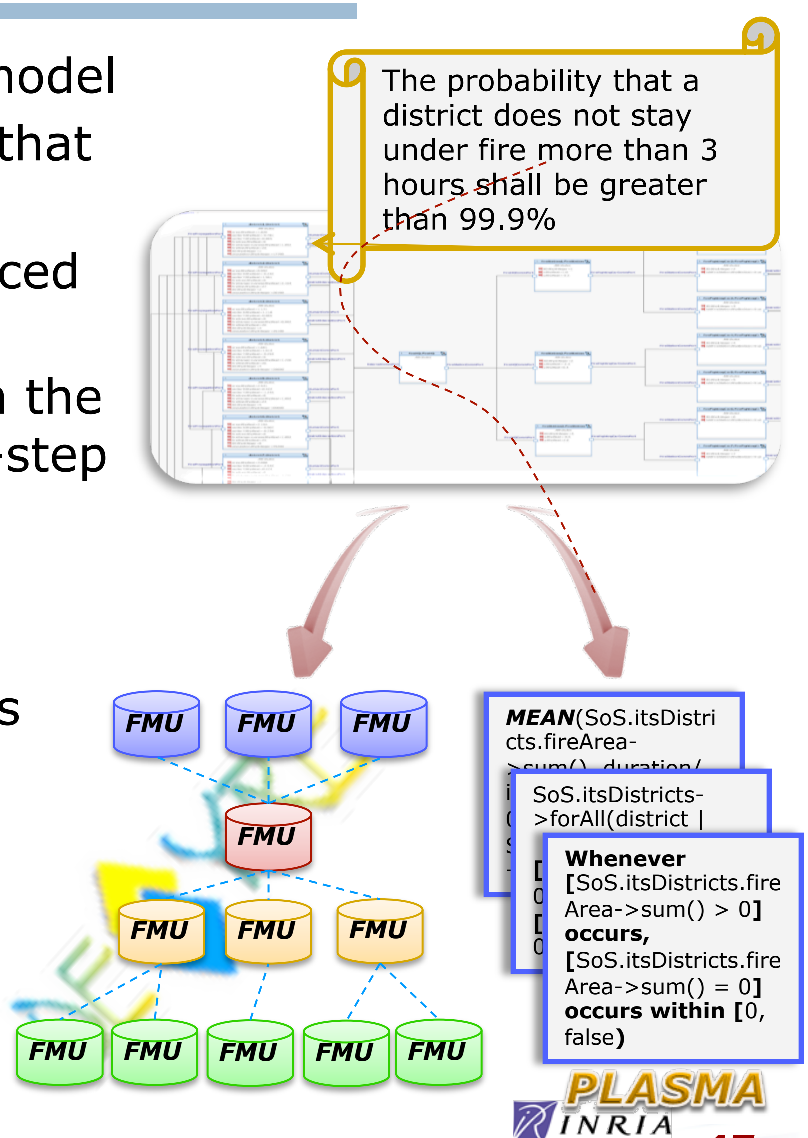
Statistical Model Checking Concepts

- Have multiple architecture alternatives, evaluated
  - Generated through different methods
  - Simulated with statistical results
  - Values for characteristics of interest
- Still must check for formal verification
  - Meet "requirements"?
  - Comply with contracts?
  - Comply with goals?
  - Note: may be many such goals/ contracts/requirements; they may conflict

Statistical Model Checking

Primary work: INRIA

- 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



SoS Emergent Behaviours

- **Emergence** definition
  - Complex pattern formation from more basic constituent parts or behaviors
  - "Properties of the whole that are not evident from the parts."

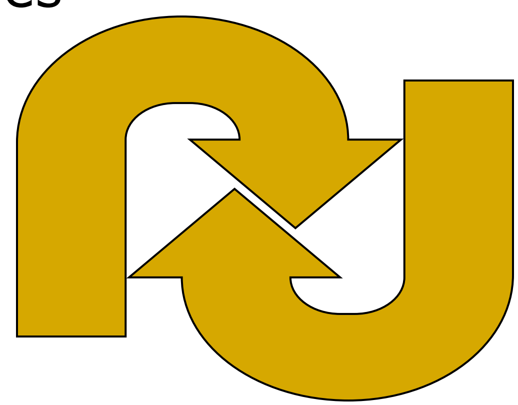
Termite cathedral mound

Classes of emergent properties

	Useful	Neutral	Destructive
Designed	**Desired capabilities**	**Facts of design**	**Accepted trade-offs**
Surprise	**Exploitable features**	**Facts of existence**	**Fearful features**

Classical ways to work with
Emergent Properties

- Top-down design
  - Identify the desired emergent properties
  - Architect the SoS – component systems and their relationships – to create the desired emergence
  - Identify the acceptable trade-offs
- Bottoms-up integration
  - Simulate and test to find the emergent properties
  - Test for presence of the desired properties
  - Test to discover the surprise properties
  - Evaluate the acceptable trade-offs



Four Methods to Check Emergence

Method	Discovery	Modelling	Analysis
Parameter-based	Observation of parameter variation	Define parameters in the UPDM model	Simulate, observe parameter behaviour, identify anomalies, emergence
Event-based	None	Model GCSL assertions of the desired events	Simulate, determine statistical compliance
Scenario-based	Observe differences in scenario behaviour from expected	Model expected scenarios Create UPDM models of the SoS	Simulate, generate traces, compare traces with the expected scenarios
Test-based	Inspect behaviour over a test objective	Create GCSL test objectives in the UPDM models	Simulate for the test case, check compliance

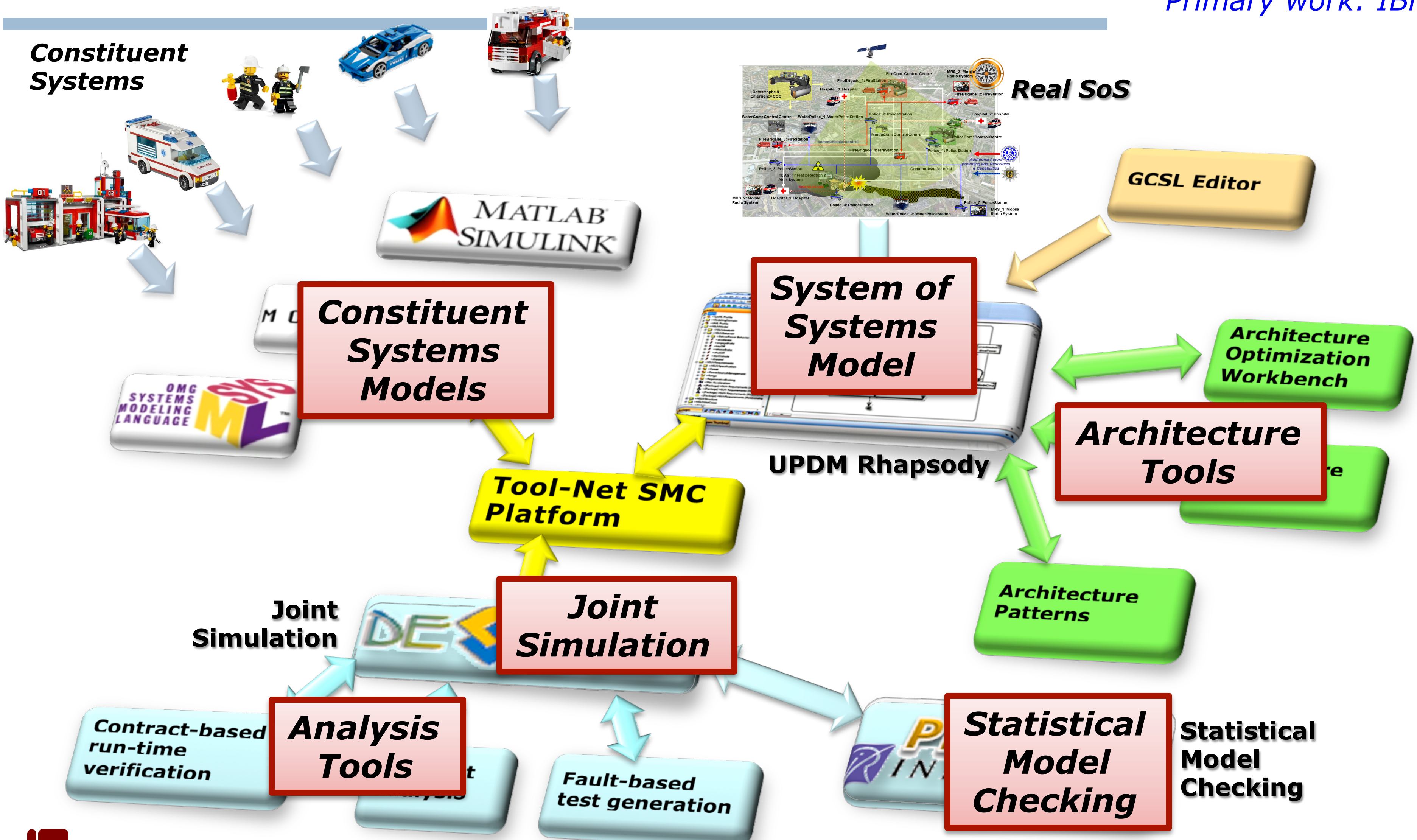
Solution Methods

DANSE Tool-Net

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

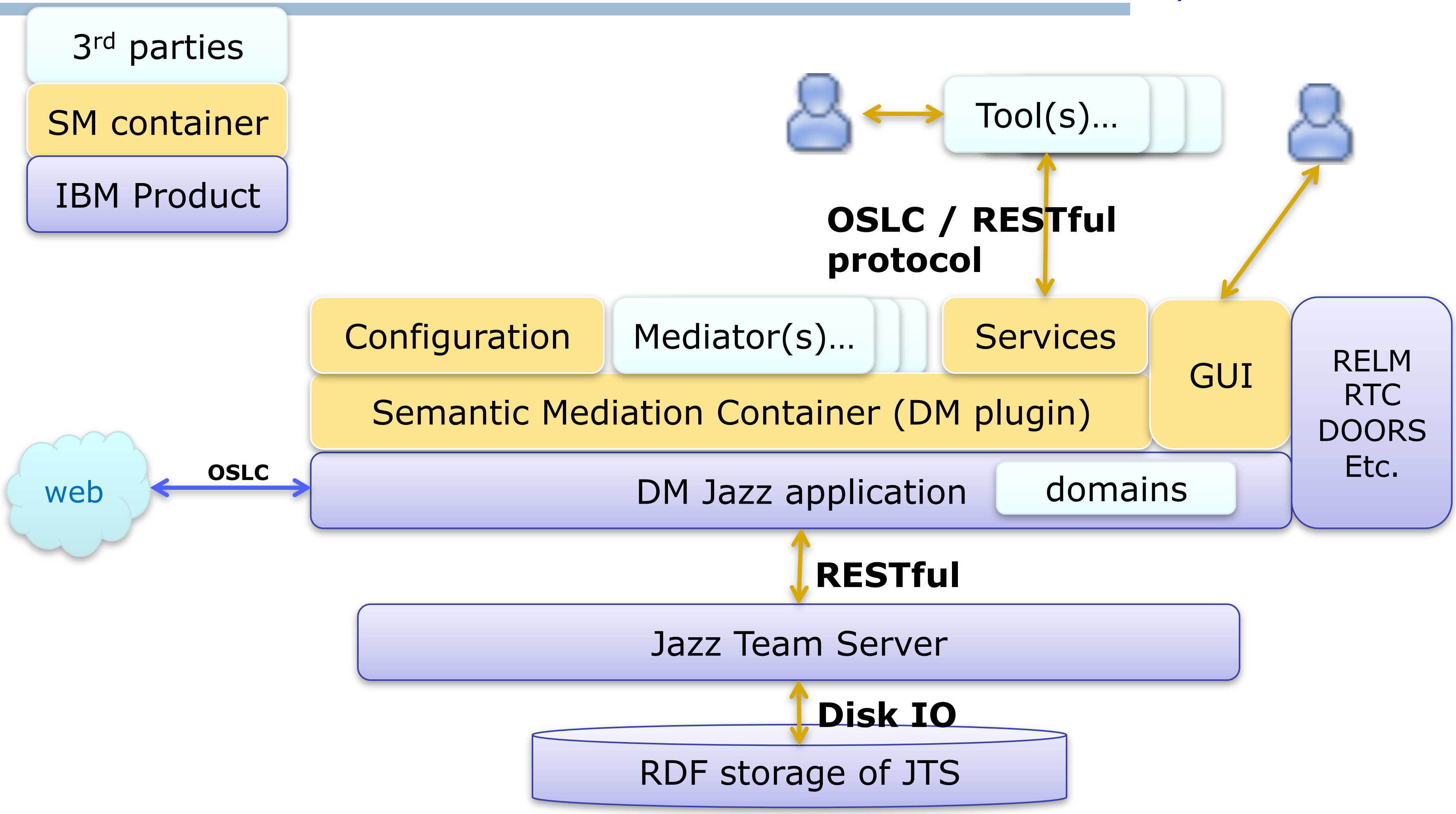
Tool-Net Connections

Primary work: IBM



Tool-Net Structure

Primary work: IBM/Sodius



Designing for **A**daptability and evolutio**N** in **S**ystem of systems **E**ngineering

Implementation

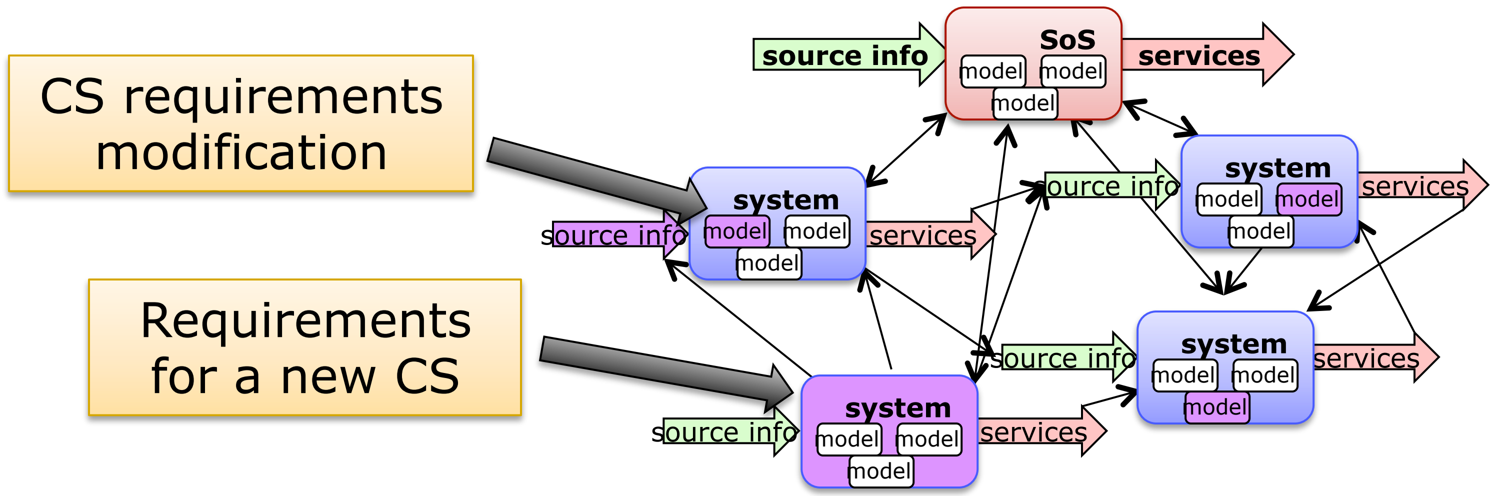
How does the DANSE methodology support change in the SoS?

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

Constituent System Requirements

- Each change to the SoS and constituent system <u>models</u> implies a change to the actual constituent systems



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

Designing for **A**daptability and evolutio**N** in **S**ystem of systems **E**ngineering

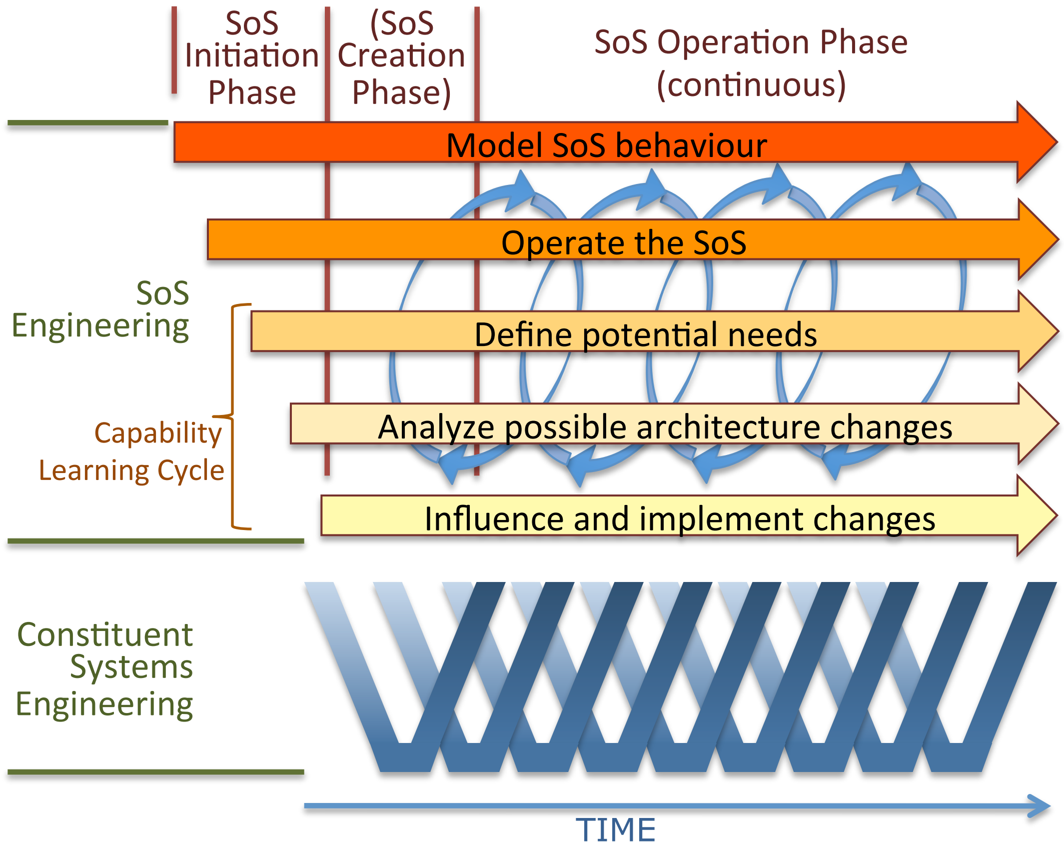
Summary

An effective methodology for SoS evolution supported by useful tools

DANSE SoS Lifecycle

Single model to embody the integrating thoughts

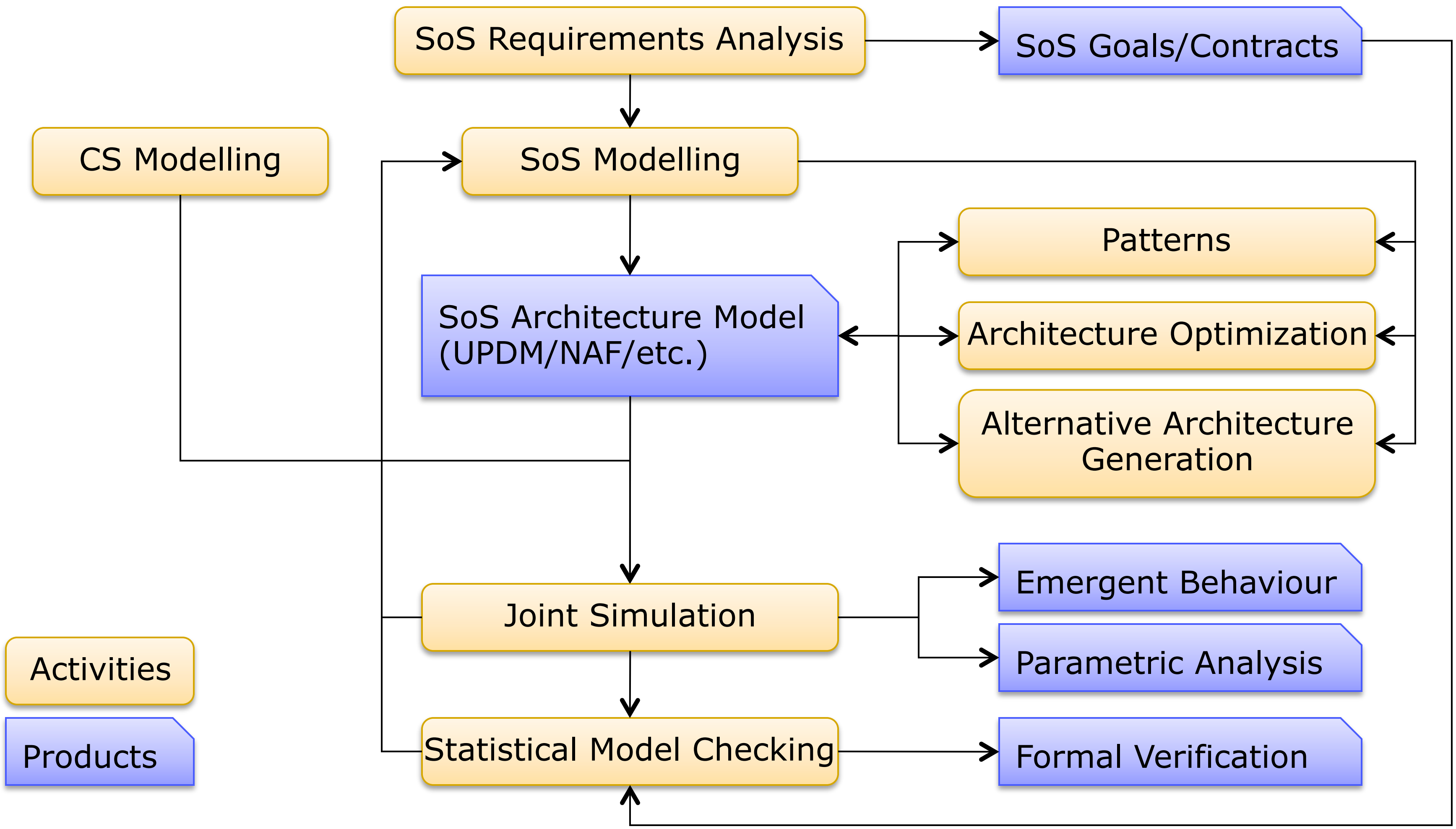
- An initiation phase
- Optional creation phase
- Forward movement through the SoS life
- Constant cycling of events/ scenarios
- A "capability learning cycle"
  - *Where the DANSE benefit happens!*
- Normal Vee-based SE in the constituent systems



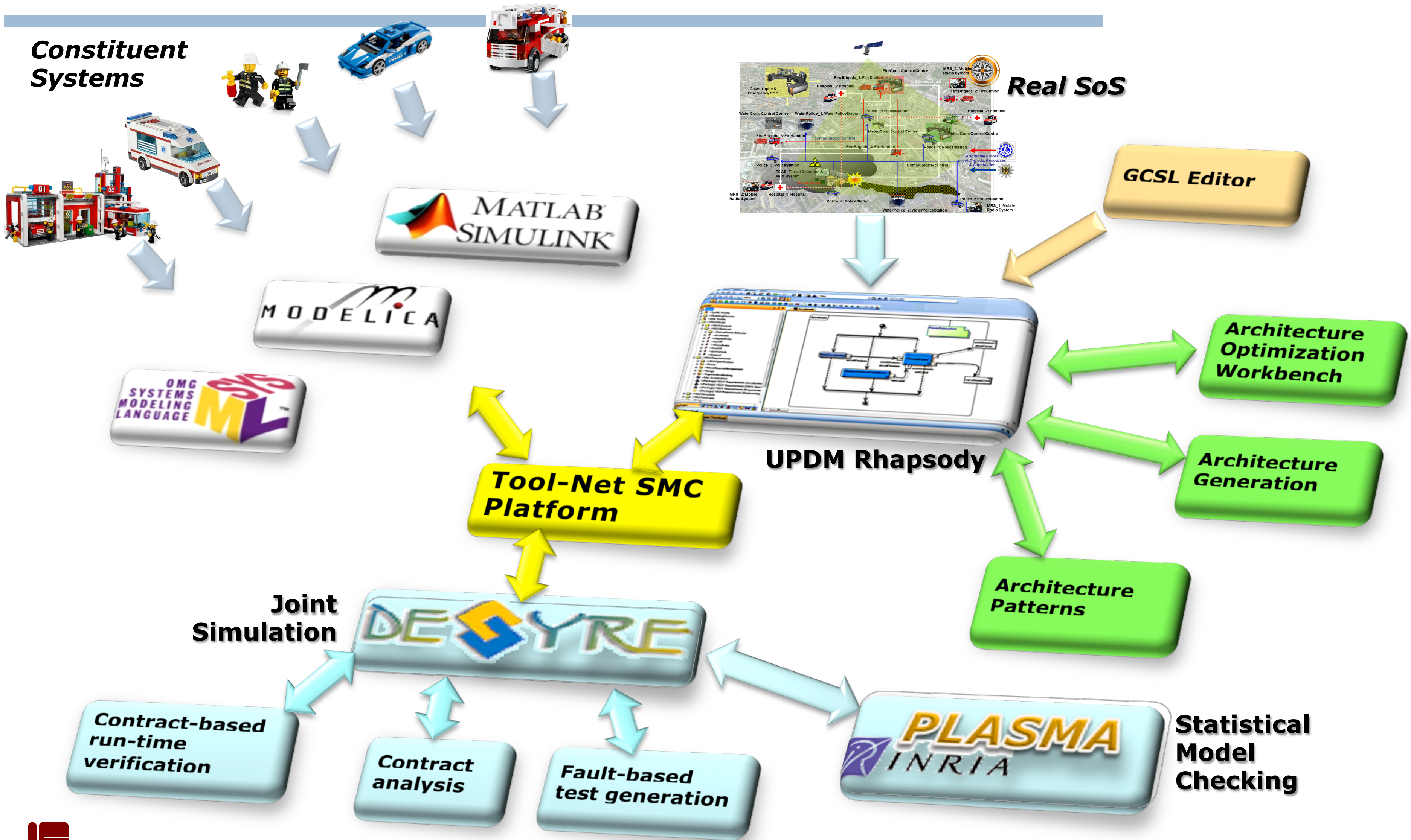
Alternate starting points:

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

Example "Use Case" of Methodology



DANSE Tools



Constituent Systems
MATLAB SIMULINK
MODELICA
OMG SYSTEMS MODELING LANGUAGE SysML
Real SoS
GCSL Editor
UPDM Rhapsody
Architecture Optimization Workbench
Architecture Generation
Architecture Patterns
Tool-Net SMC Platform
Joint Simulation
DESYRE
Contract-based run-time verification
Contract analysis
Fault-based test generation
PLASMA INRIA
Statistical Model Checking

DANSE

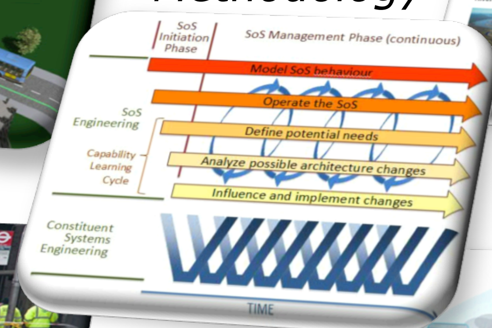
Eric Honour
+1 (615) 614-1109
ehonour@hcode.com

Designing for **A**daptability and evolutio**N** in **S**ystem of systems **E**ngineering

Automated Ground Transport *Carmeq*

IAI *Integrated Water Treatment and Supply*

DANSE Methodology



Emergency Response *Airbus*

Thales *Air Traffic Management*

SEVENTH FRAMEWORK PROGRAMME