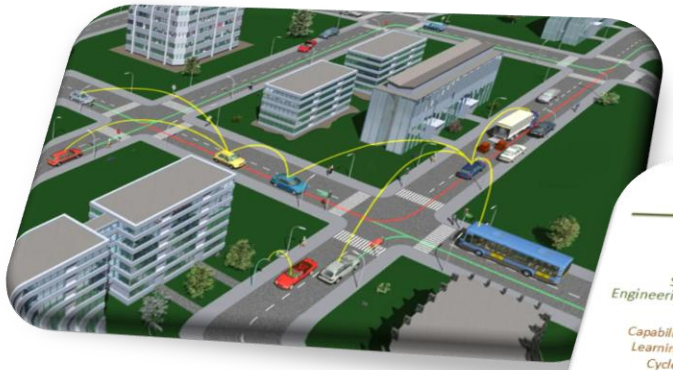
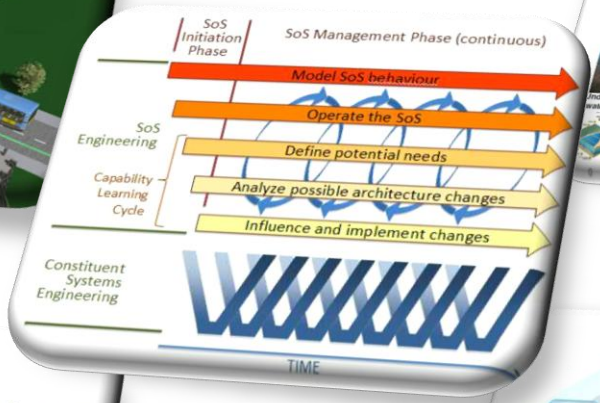
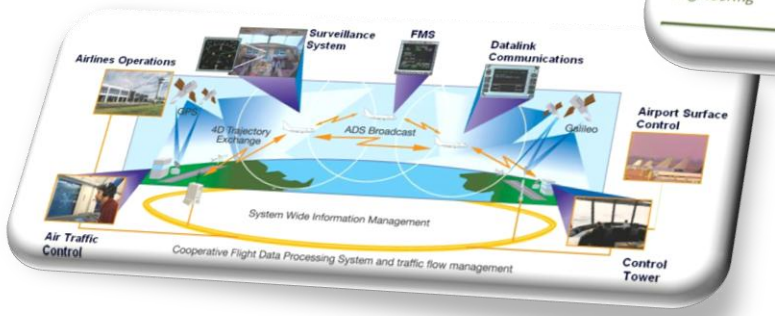
Honourcode, Inc.
Eric Honour
+1 (615) 614-1109
ehonour@hcode.com

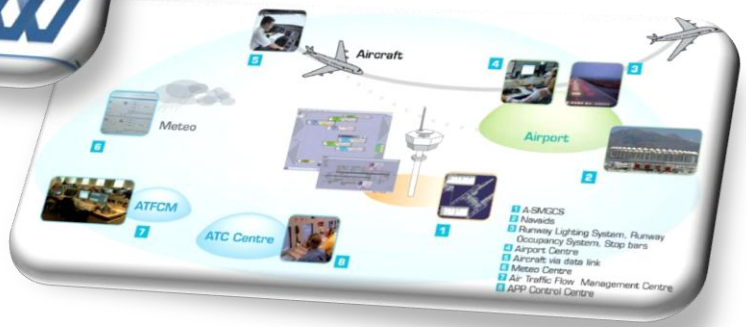
DANSE

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









SEVENTH FRAMEWORK PROGRAMME

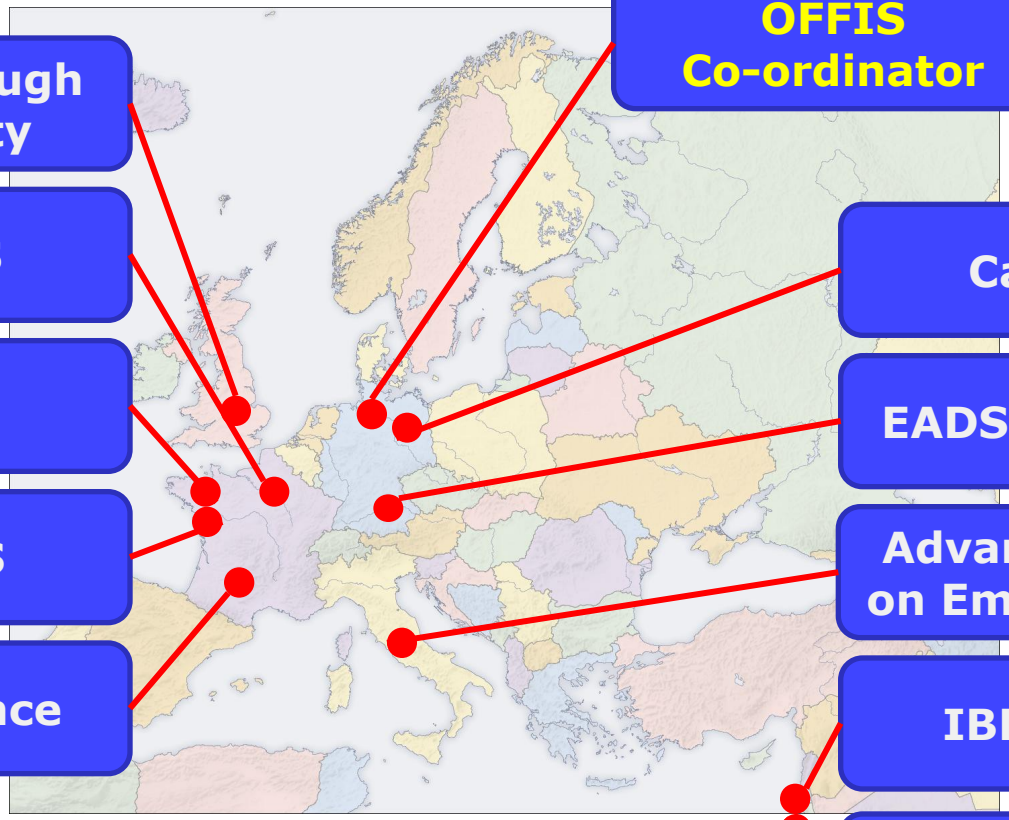
Topics

- DANSE consortium and goals
- Systems of systems concepts
- DANSE methodology
- DANSE technologies
  - SoS/CS modeling
  - Goals and contracts specification language (GCSL)
  - Automated architecture generation
  - Joint simulation
  - Statistical model checking

DANSE in a Nutshell

- Develop approaches for SoS engineering (design + manage)
  - *Methodology* to support evolution, adaptive and iterative SoS life-cycle
  - *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 Service; Air Traffic Management; Autonomous Ground Transport; Integrated Water Treatment and Supply
- Exploitation & dissemination of SoS technology

DANSE Consortium



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

Contact:
Bernhard Josko
josko@offis.de

SEVENTH FRAMEWORK PROGRAMME

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?

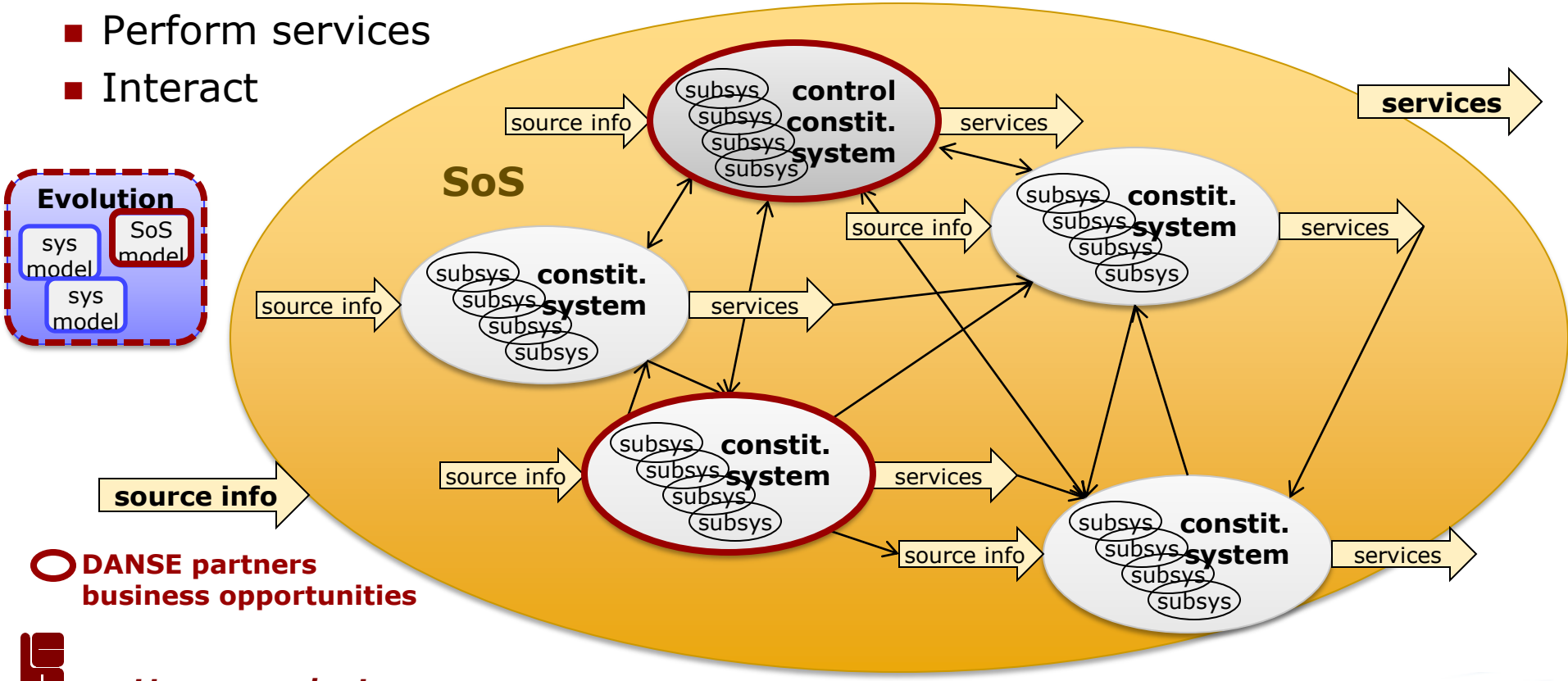
Architecture of an SoS

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



DANSE partners business opportunities

SoS Challenges

Management Challenges

- Size and scope unmanageable by a single entity
- Lack of central specification/development/control
- Management can overshadow engineering
- Fuzzy boundaries cause confusion
- Constituent systems have different life cycles

Technical Challenges

- Ambiguous, constantly changing "requirements"
  - So verification/validation present problems
- Constituent systems constantly changing
- Complexity is a major issue, resulting in surprise emergent behavior
- SoS engineering is evolutionary, with no clear start and no clear finish

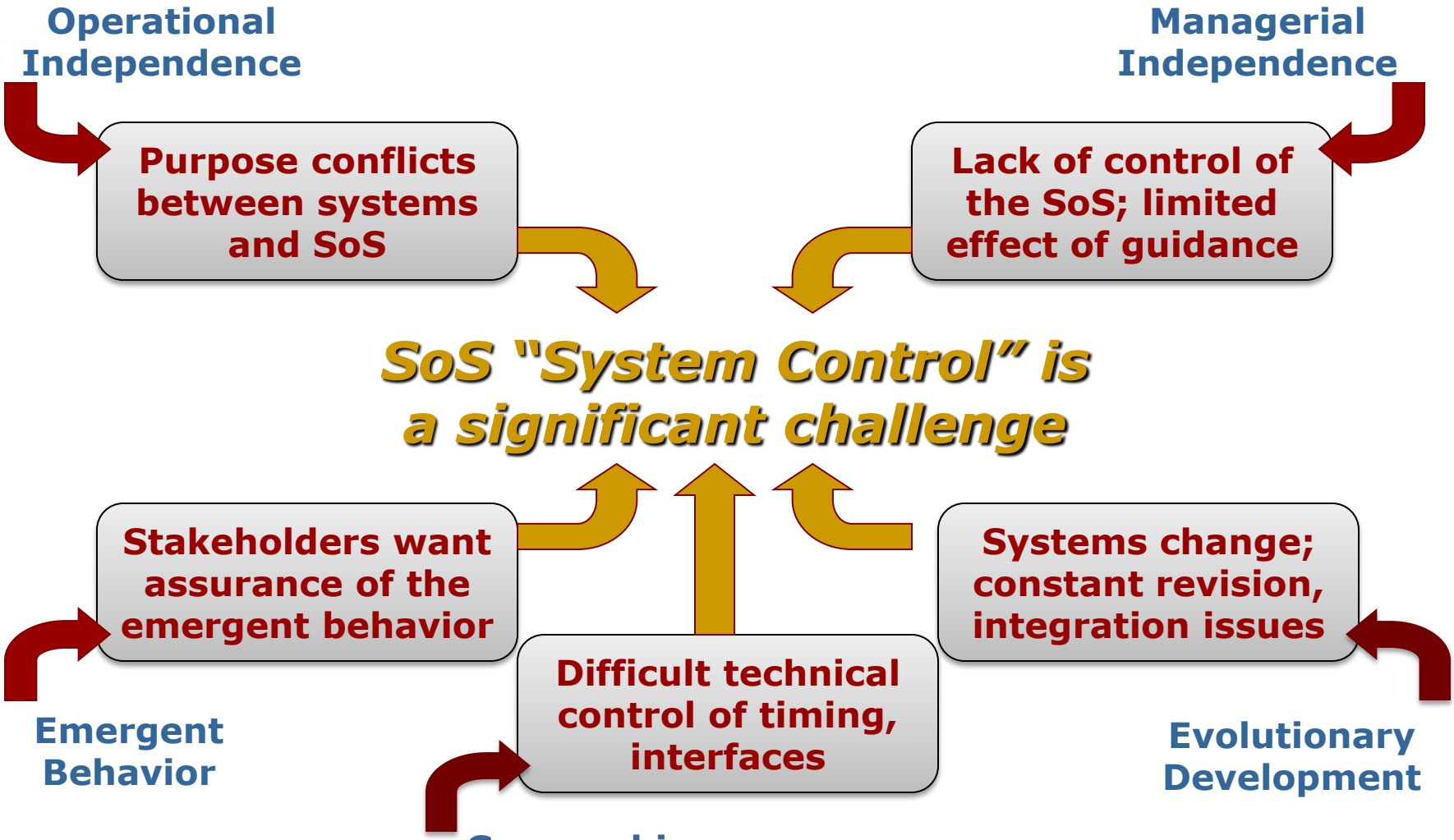
SoS Characteristics

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

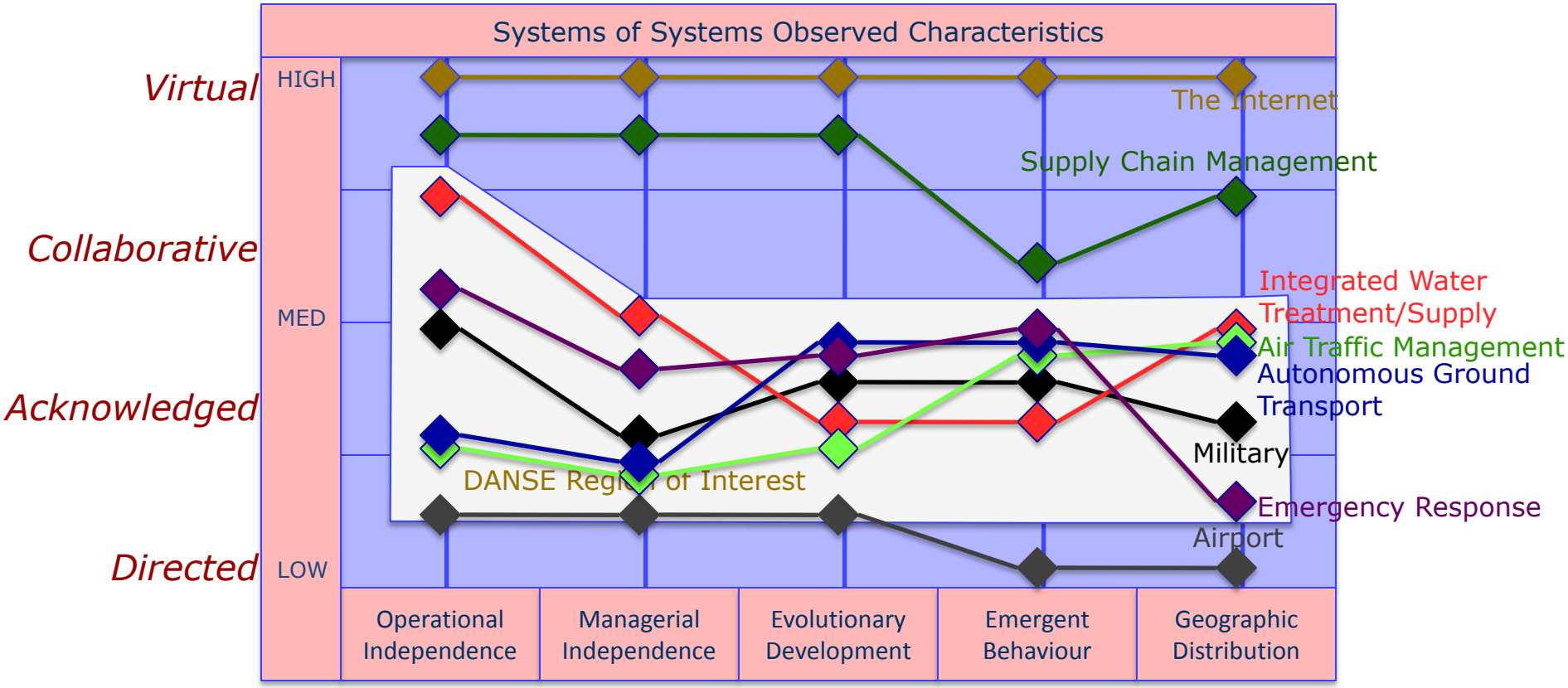
- Emergent behavior - SoS performs functions not achievable by the independent component systems
- Geographic distribution - geographic extent forces the elements to exchange information in a remote way
- Evolutionary development - functions and purposes are added, removed and modified in an ongoing 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)

Impacts of SoS Characteristics

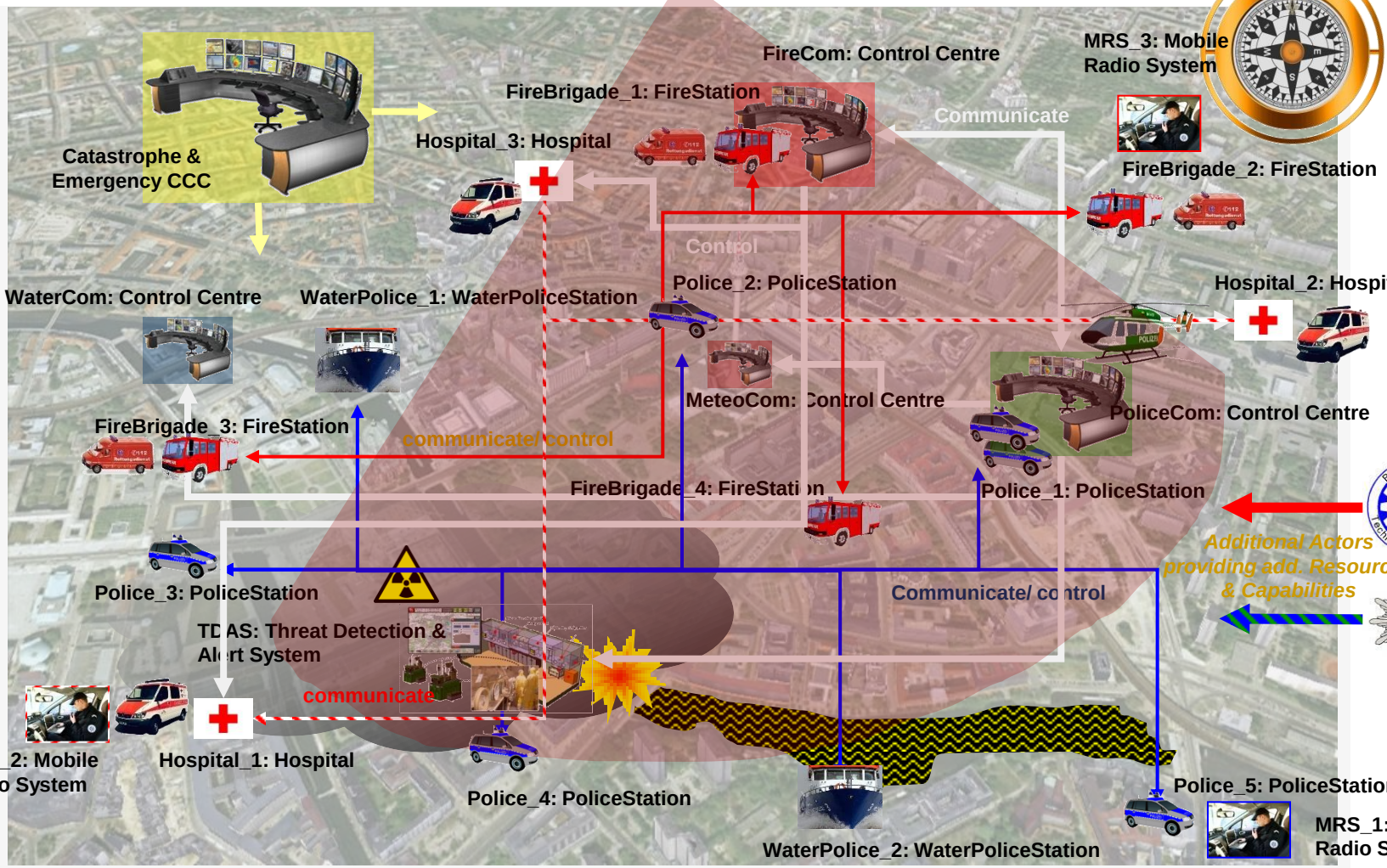


Differing Levels of "SoS-ness"



Emergency Response SoS

Slide by Tim Lochow, EADS



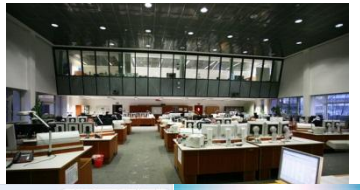
Dynamicity in the ER SoS

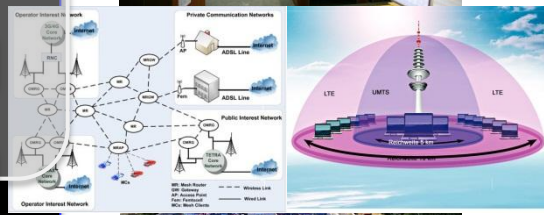
Slide by Tim Lochow, EADS

SoSE Challenges

SoS operational timeline and dynamicity aspects

	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			





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

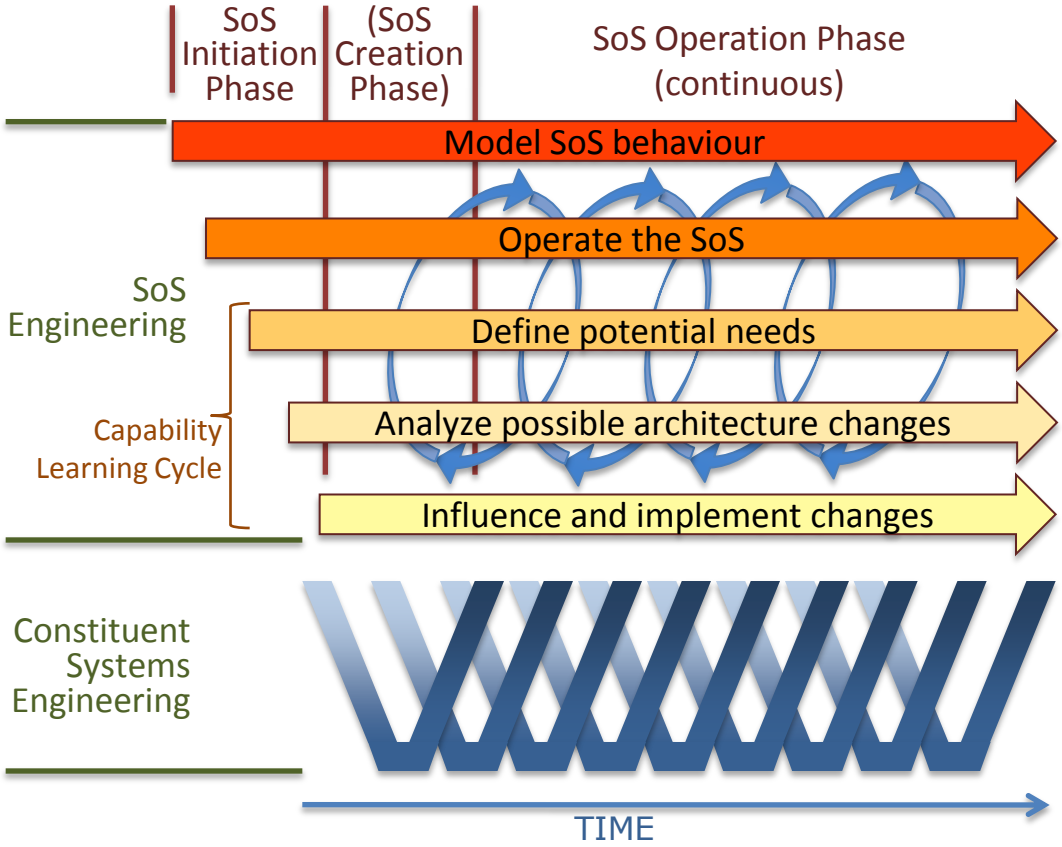
DANSE Methodology Overview

What is the DANSE methodology, and how does it integrate the methods and 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



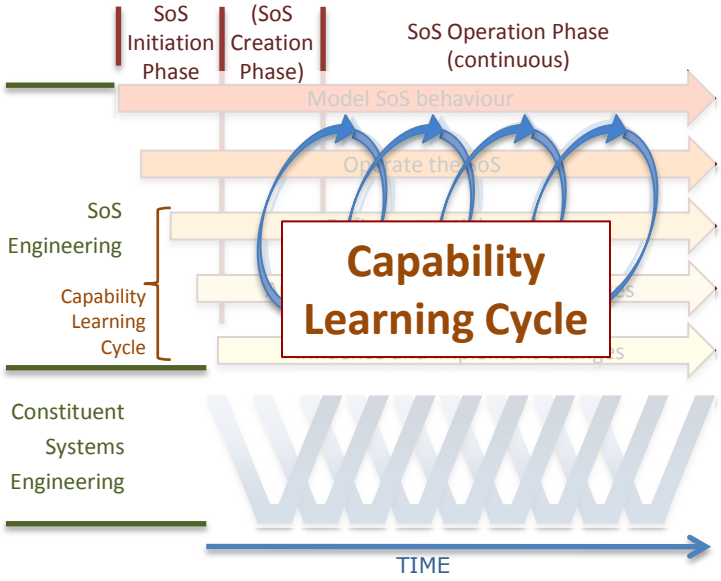
Alternate starting points:

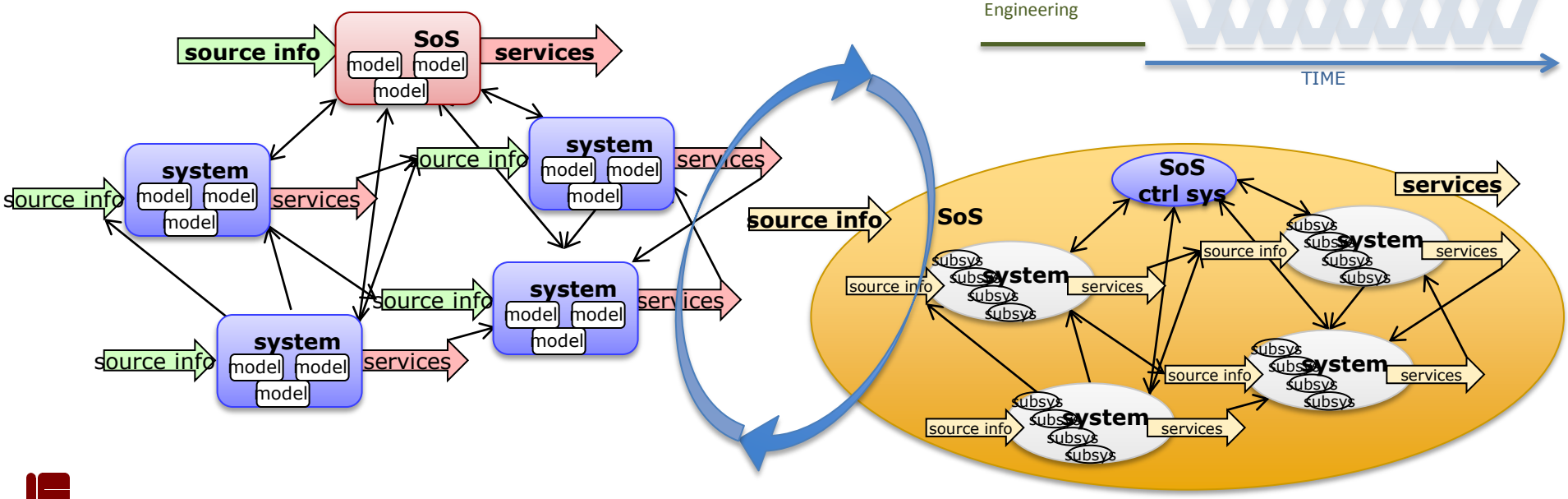
- SoS is acknowledged among existing systems
- SoS is created by a Lead System Integrator

SoS Operation Phase

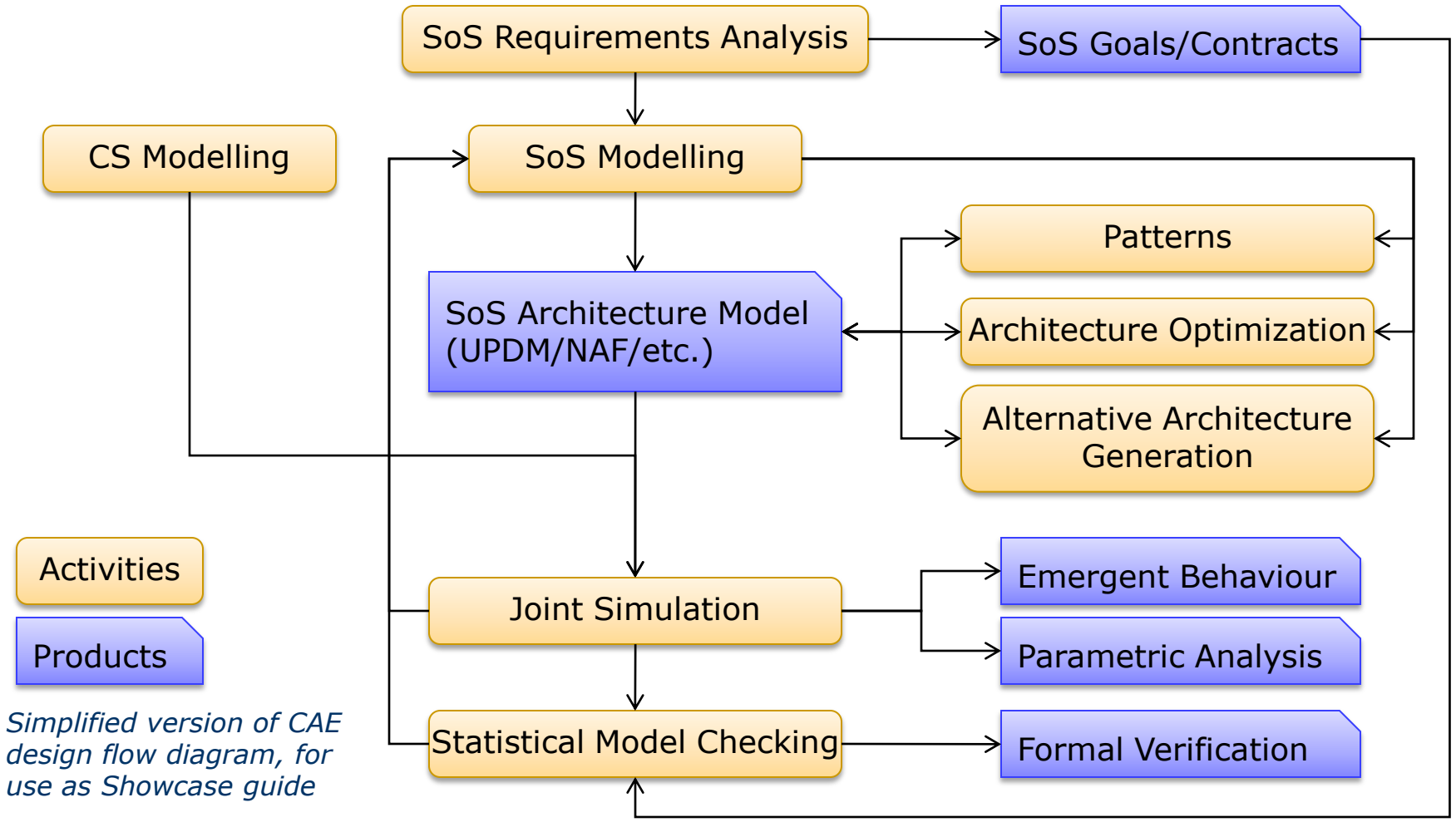
Capability Learning Cycle

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





Example "Use Case" of Methodology



Simplified version of CAE design flow diagram, for use as Showcase guide

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

DANSE Technologies

What technology break-throughs are coming from DANSE, and how do they work in the methodology?

Technology Topics

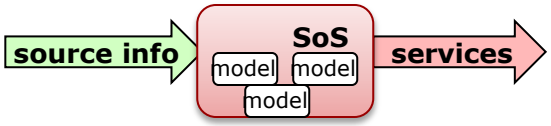
- SoS/CS modeling
- Goals and contracts specification language (GCSL)
- Automated architecture generation
- Architecture patterns
- Graph grammar
- Concise modeling
- Joint simulation
- Statistical model checking

SoS Models

- SoS models take many forms, serve different purposes:
  - Behavioral – observe/predict SoS behavior
  - Performance – predict parameter values
  - Specialty – analyze characteristics of interest
- SoS models create information that is NOT available from the constituent system models
- ***Most complete forms use architectural frameworks – for DANSE, assumed form is UPDM***

Emergency SoS Examples

- *Resource dispatching effectiveness model*
- *Site resources physical location model*
- *Call handling throughput model*
- *Electromagnetic compatibility model*
- *Total threat response model*

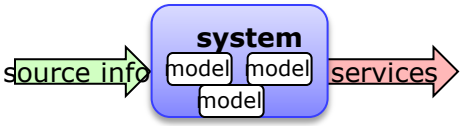


Constituent System Models

- Created during engineering development of the constituent system, for different purposes than SoS
  - Constituent system bounds
  - Constituent system goals
  - Different modeling goals
  - Different characteristics of interest
- May be multiple models for each constituent system, in different formats (SysML, Modelica, Simulink, etc.)
- May be useful for the SoS modeling effort
  - Usually more detailed than desired for SoS
  - May have execution performance issues

Fire Brigade Examples

- *Resource tracking system behavioral model*
- *Communications system performance model*
- *Fire brigade response time model*



Goals and Contracts Spec Language (GCSL)

Provides an automation of goals and contracts that can be used in the joint simulation

The **user** works on the **syntax** level and must only be aware of the **semantics of the pattern**, not of its formal representation

The **GCSL editor translates** the textual pattern to formal representation

"All FireStation hosts at least one Fire Fighting Car"

```
SoS.itsFireStations->forAll(fstation | fstation.hostedFireFightingCars->size() >= 1)
```

"Any district cannot have more than 1 fire station, except if all districts have at least 1"

```
SoS.itsDistricts->exists(district | district.containedFireStations->size() > 1) implies
SoS.itsDistricts->forAll(containedFireStations->size() >= 1)
```

"The fire fighting cars hosted by a fire station shall be used all simultaneously at least once in 6 months"

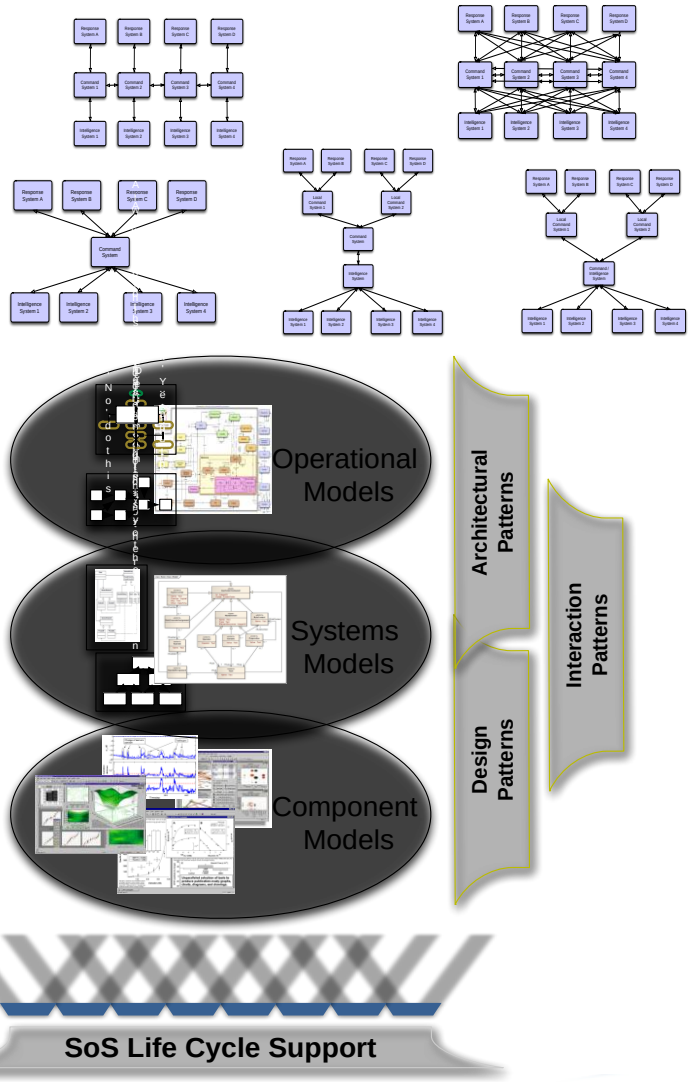
```
SoS.itsFireStations->forAll(fireStation |
  Whenever [fireStation.hostedFireFightingCars->exists(isAtFireStation)] occurs,
        [fireStation.hostedFireFightingCars->forall(isAtFireStation = false)]
                                                    occurs within [6 months])
```

Purpose of GCSL: enable automated statistical model checking against the goals and contracts

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

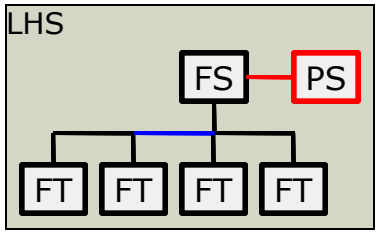
- DANSE developing an SoS pattern repository
  - Searchable database of patterns
  - UPDM profiles that can be inserted into the SoS model

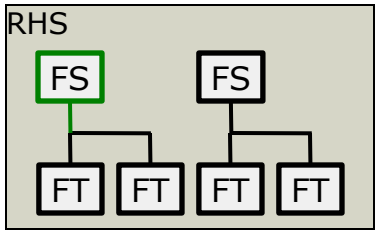


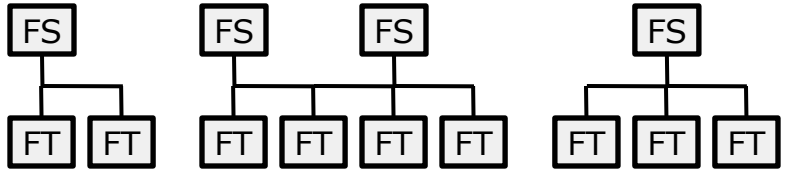
Graph Grammar

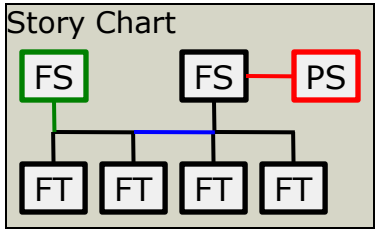
- Rules for changing the form of a set of relationships
  - **Left hand side** (LHS) depicts a pattern that can be matched
  - **Right hand side** (RHS) depicts a transformed version
  - **Story Chart** combines LHS and RHS into a transformation rule
- Any successful find of the LHS pattern can be replaced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

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

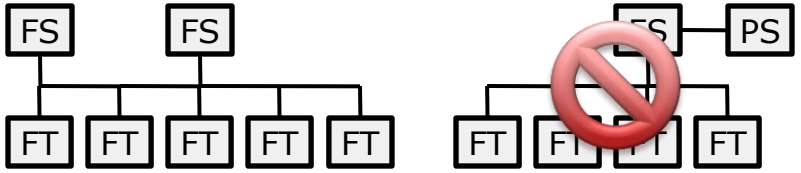








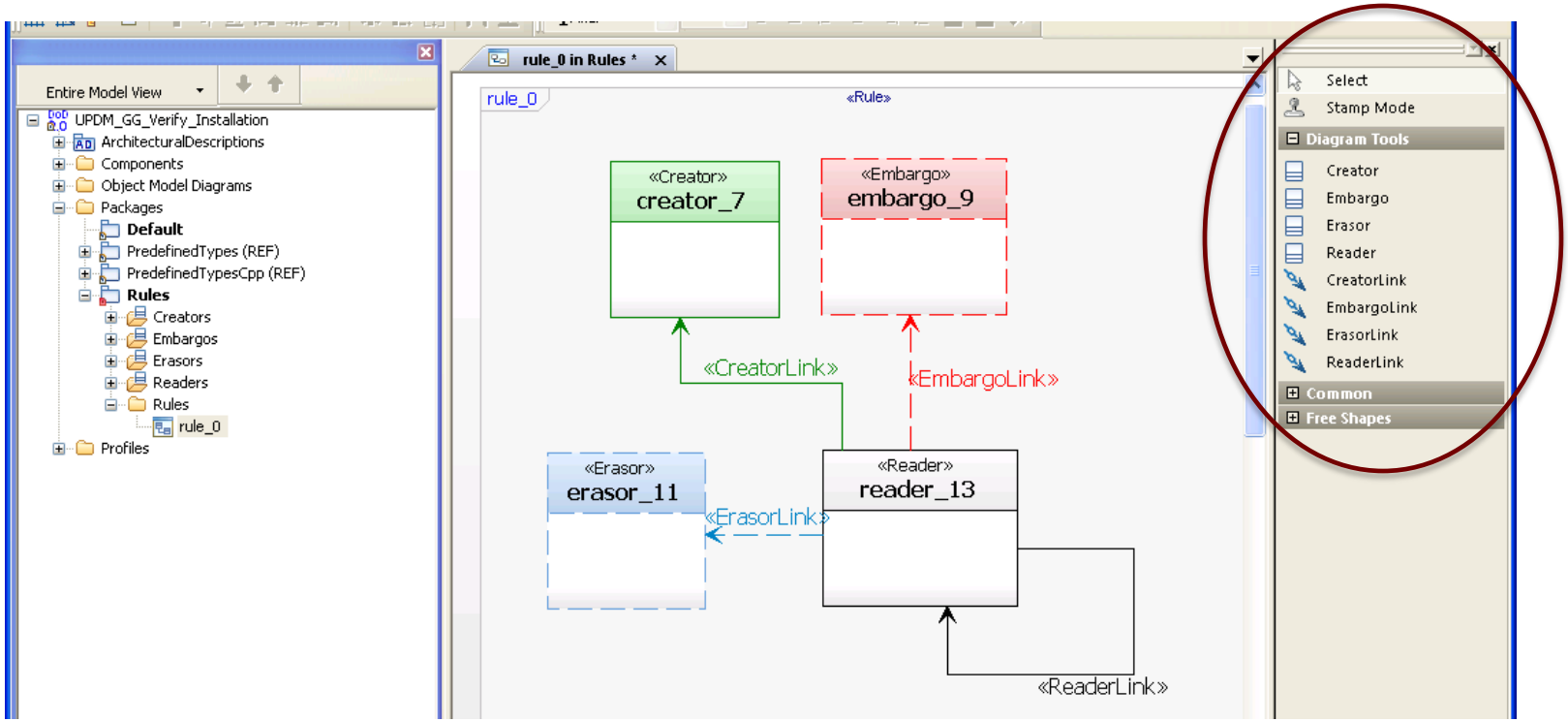
FT: Fire truck
FS: Fire station
PS: Police station



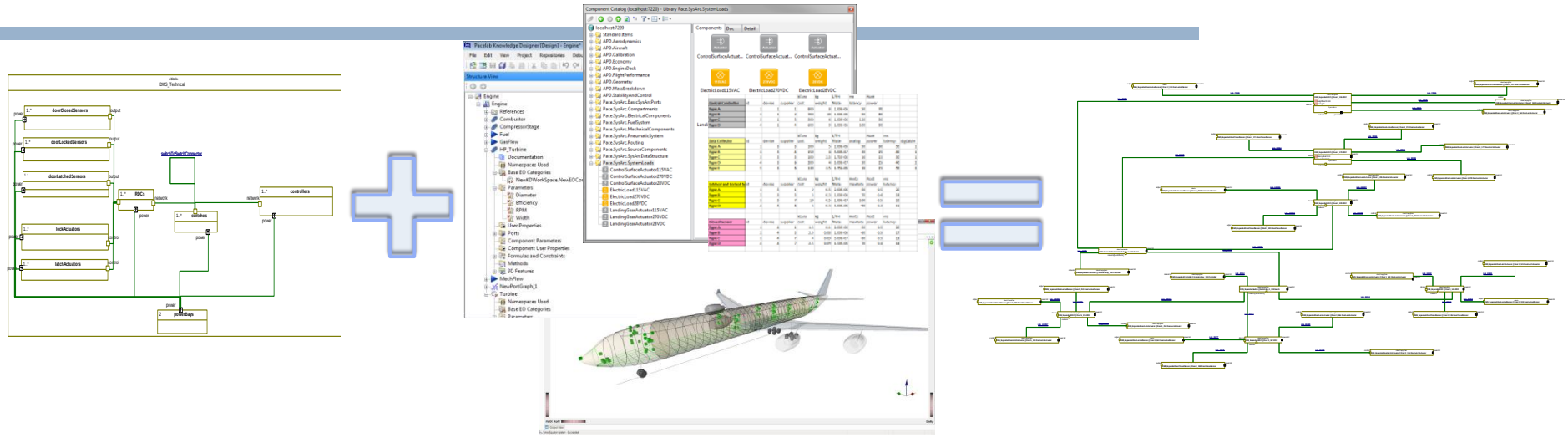
DANSE Graph Grammar

- Story Charts **implemented** as special UPDM diagrams
- Based on a UPDM **profile** to enable the modeling of a rule

→ *Dynamicity modeling is done in the same language as the modeling of the SoS itself.*



Concise Modeling



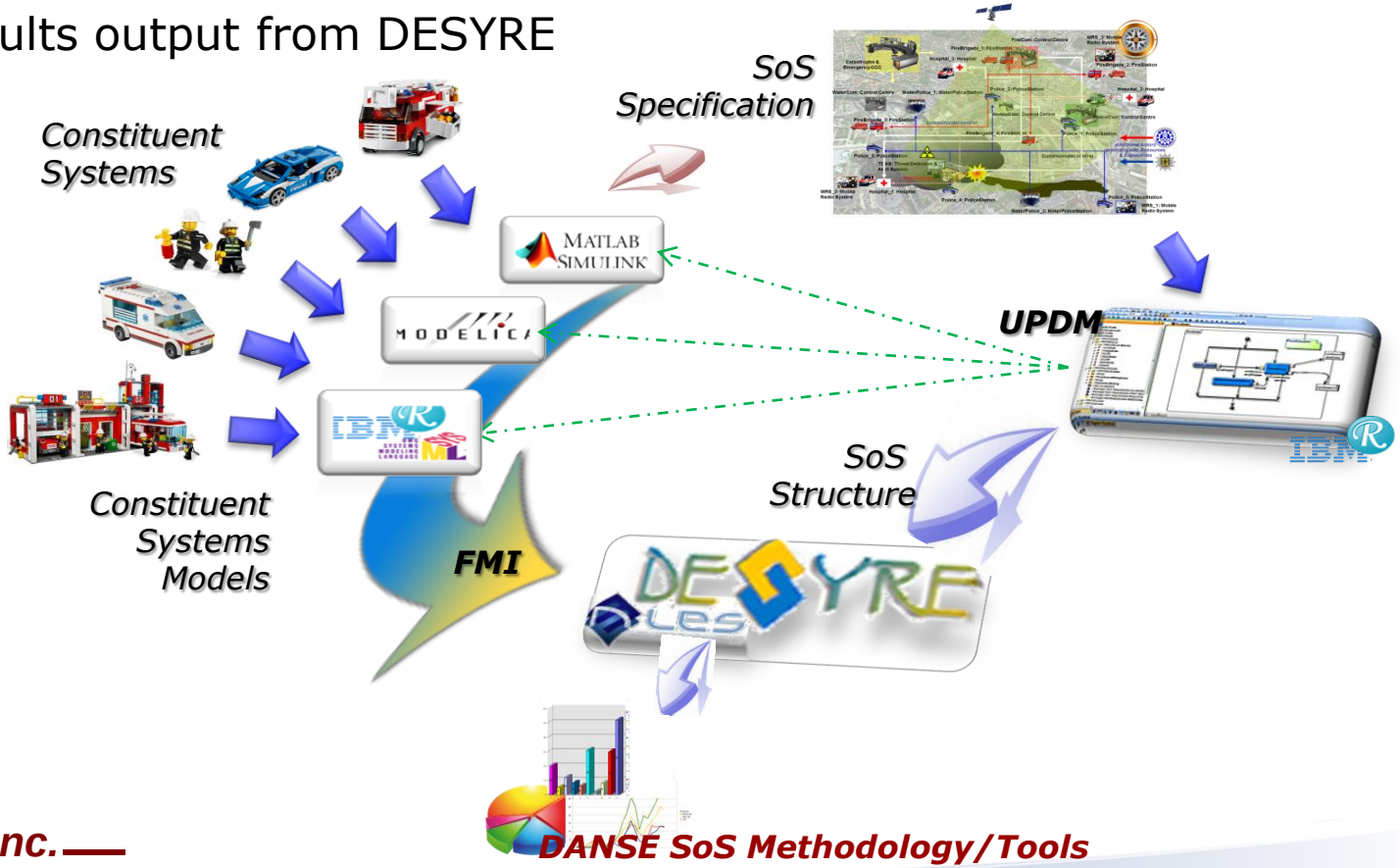
SysML models combined with tabular data

- 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

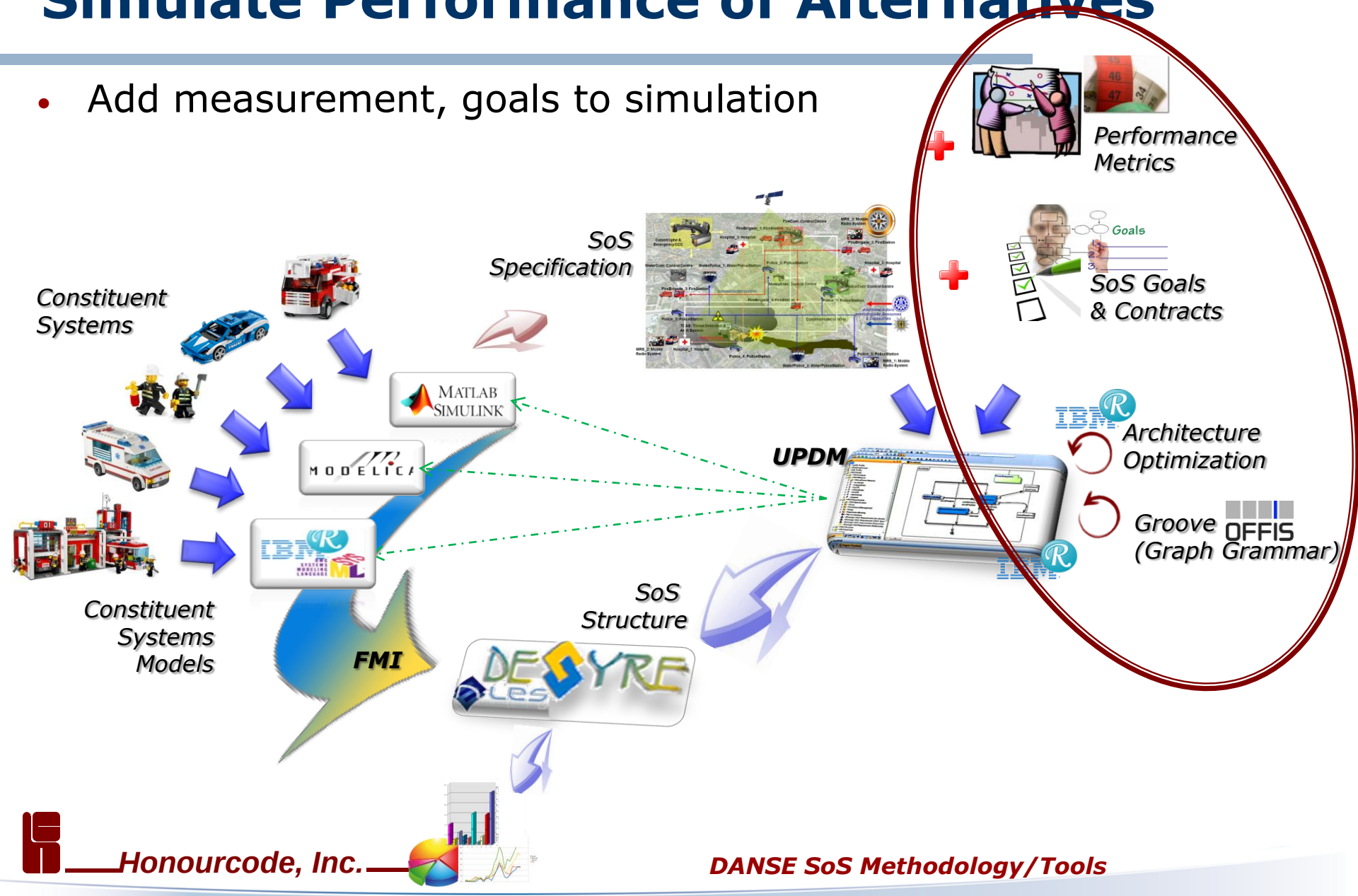
Joint simulation

- 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



Simulate Performance of Alternatives

- Add measurement, goals to simulation

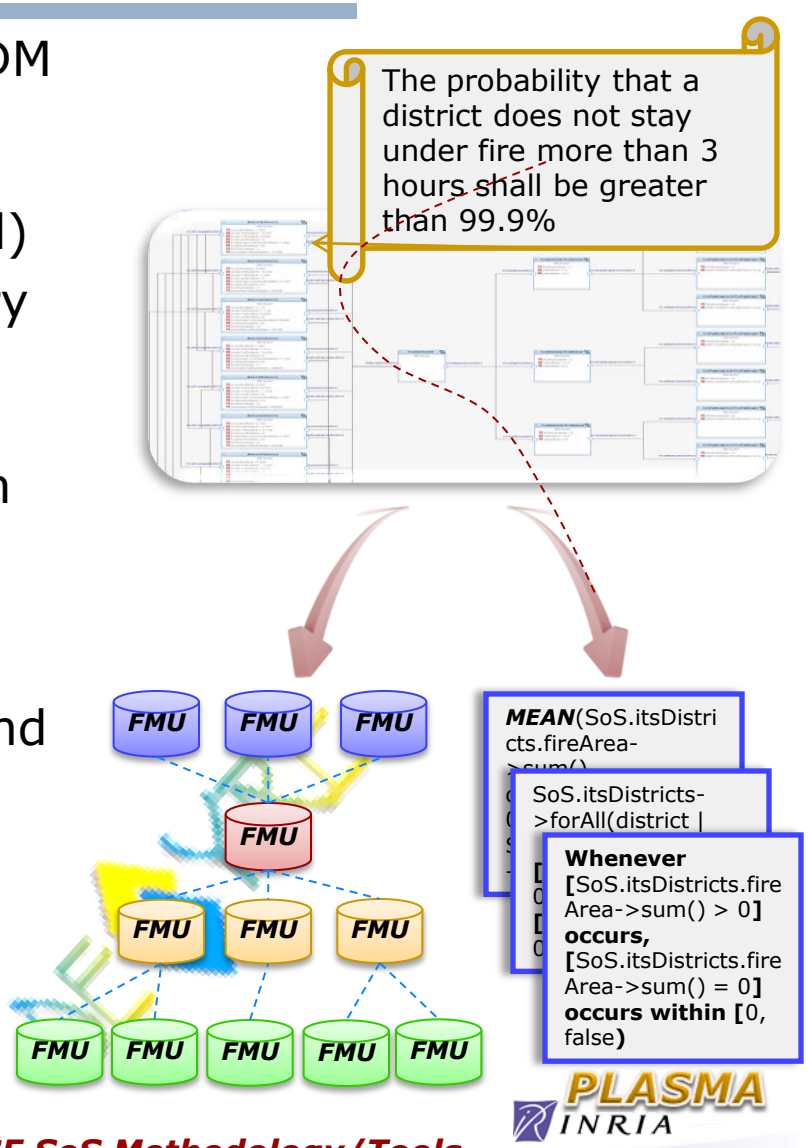


Statistical Model Checking

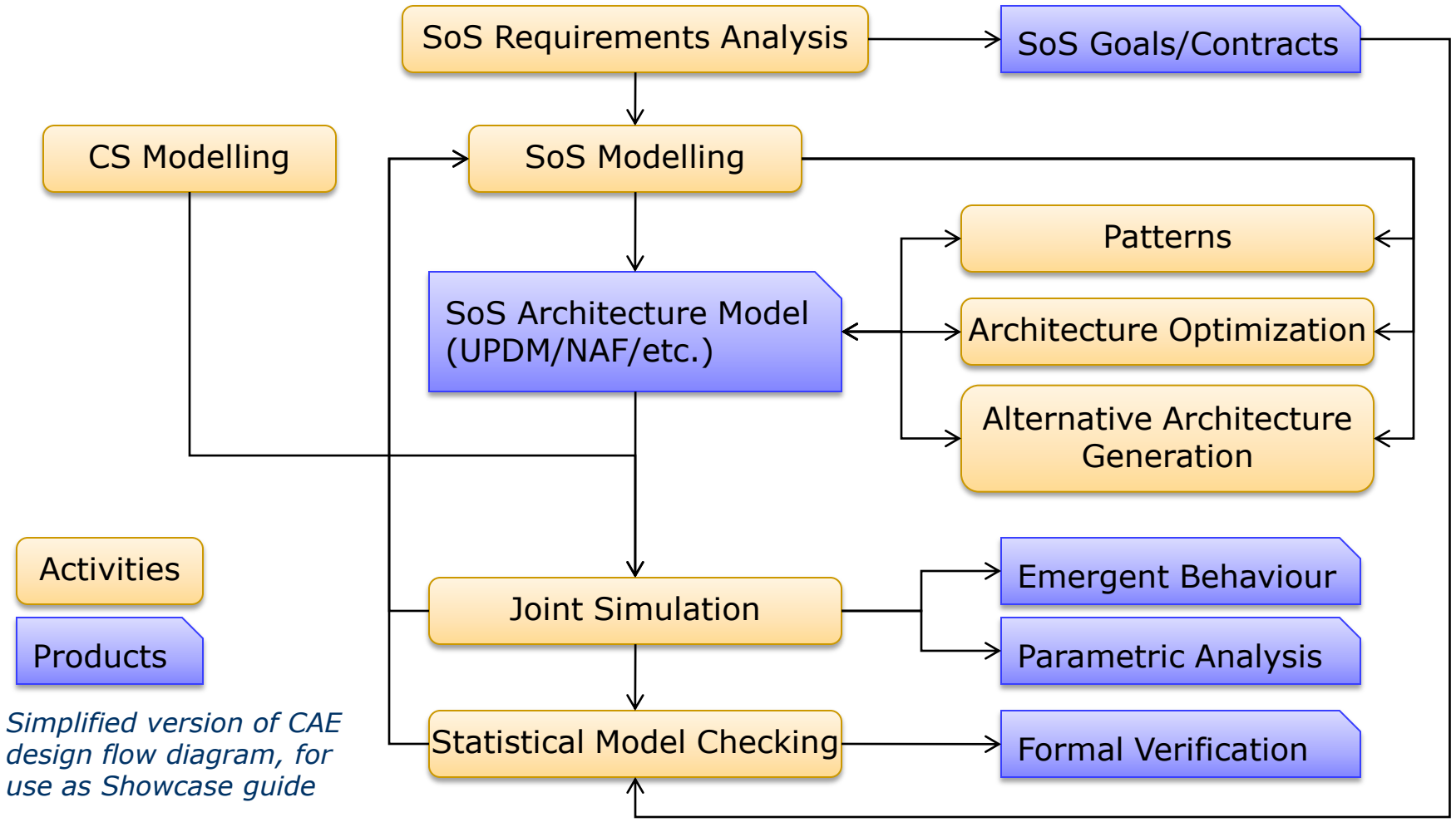
- Want to evaluate multiple architecture alternatives
  - Generated through different methods
  - Simulated with statistical results
  - Values for characteristics of interest
- Joint simulation allows
  - Performance calculation
  - Observing emergent behaviors
- Still must check for formal verification
  - Comply with contracts?
  - Comply with goals?
  - Note: may be many such goals/contracts; they may conflict

Statistical Model Checking

- Goals and Contracts specified in the UPDM model are translated into a set of properties that can be evaluated by PLASMA (Statistical Model Checking Tool)
- UPDM model variables that are necessary to evaluate the properties are set as *observable* and traced by the simulator
- DESYRE simulator provides PLASMA with the variable values step-by-step during the simulation
- PLASMA verifies the properties and returns the Statistical Model Checking and Contract verification results



Example "Use Case" of Methodology

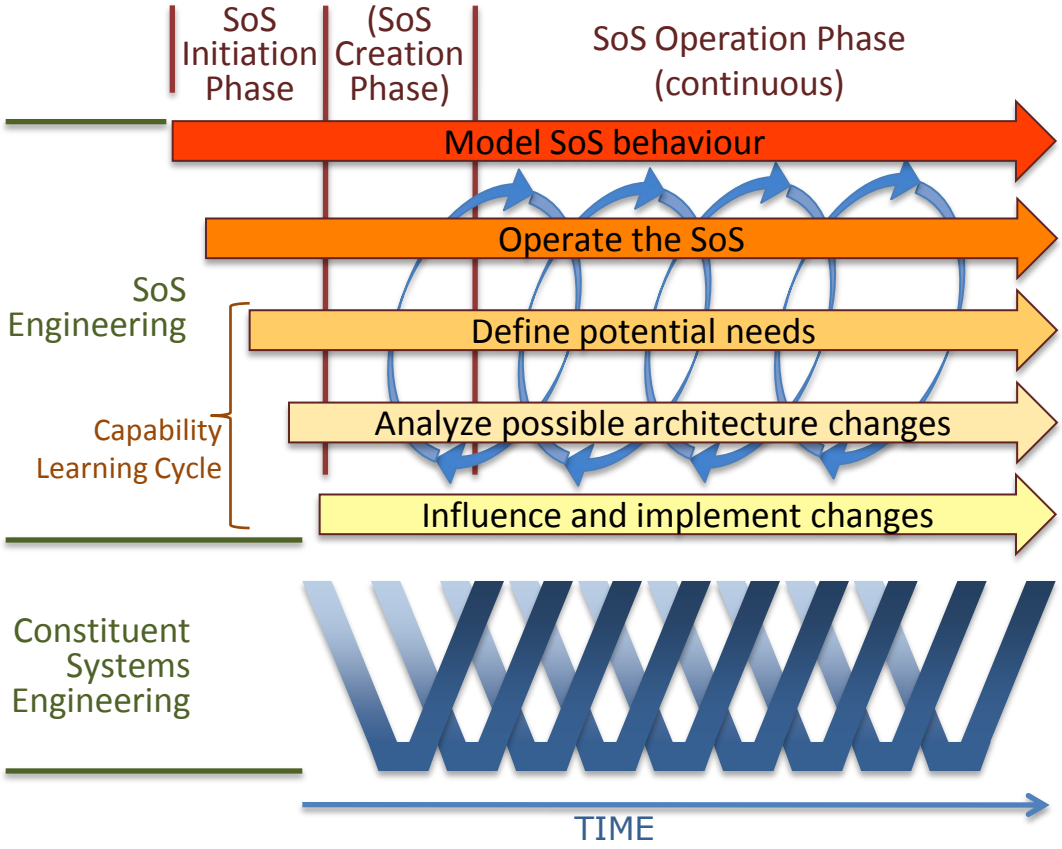


Simplified version of CAE design flow diagram, for use as Showcase guide

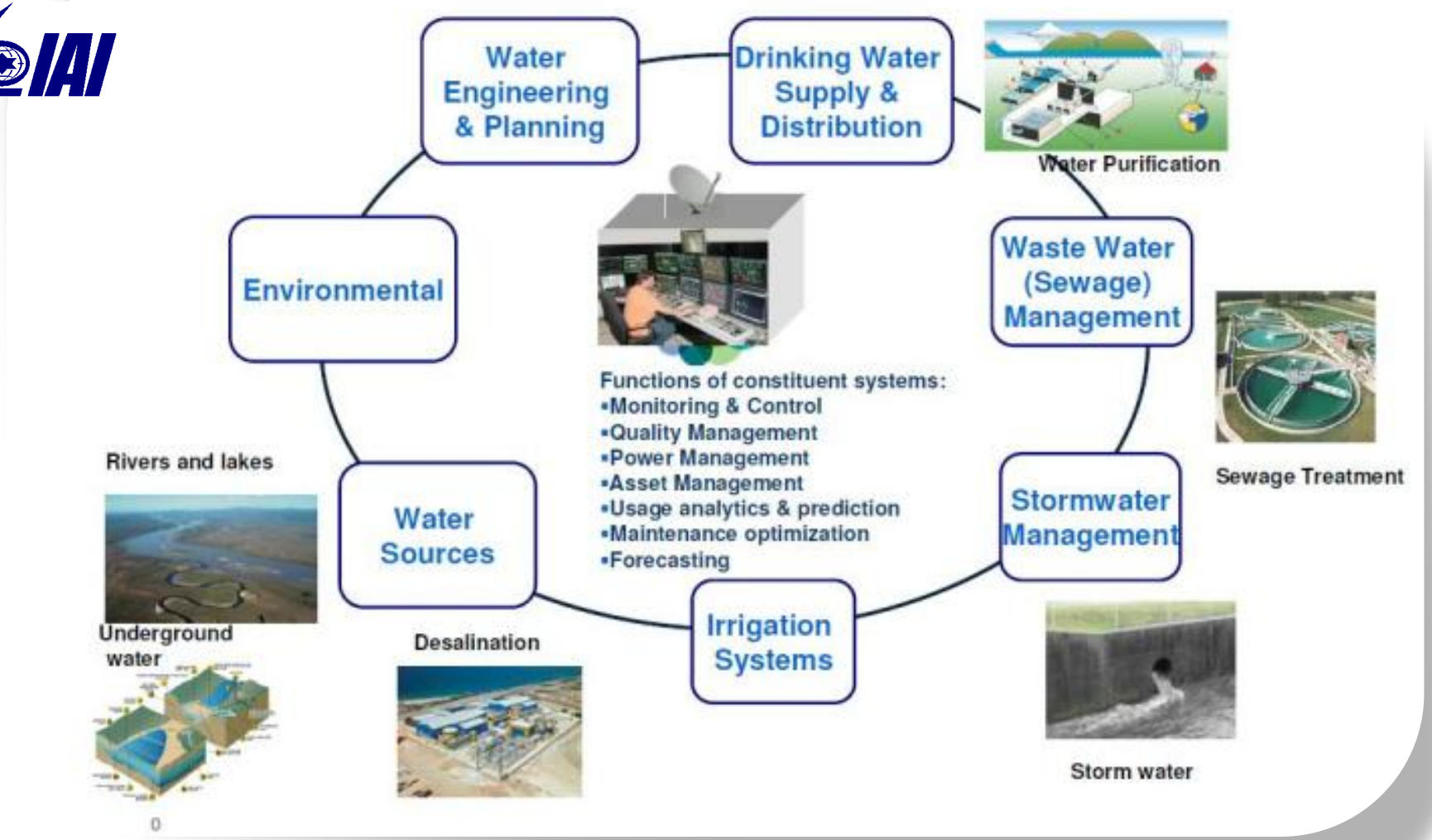
DANSE Methodology

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



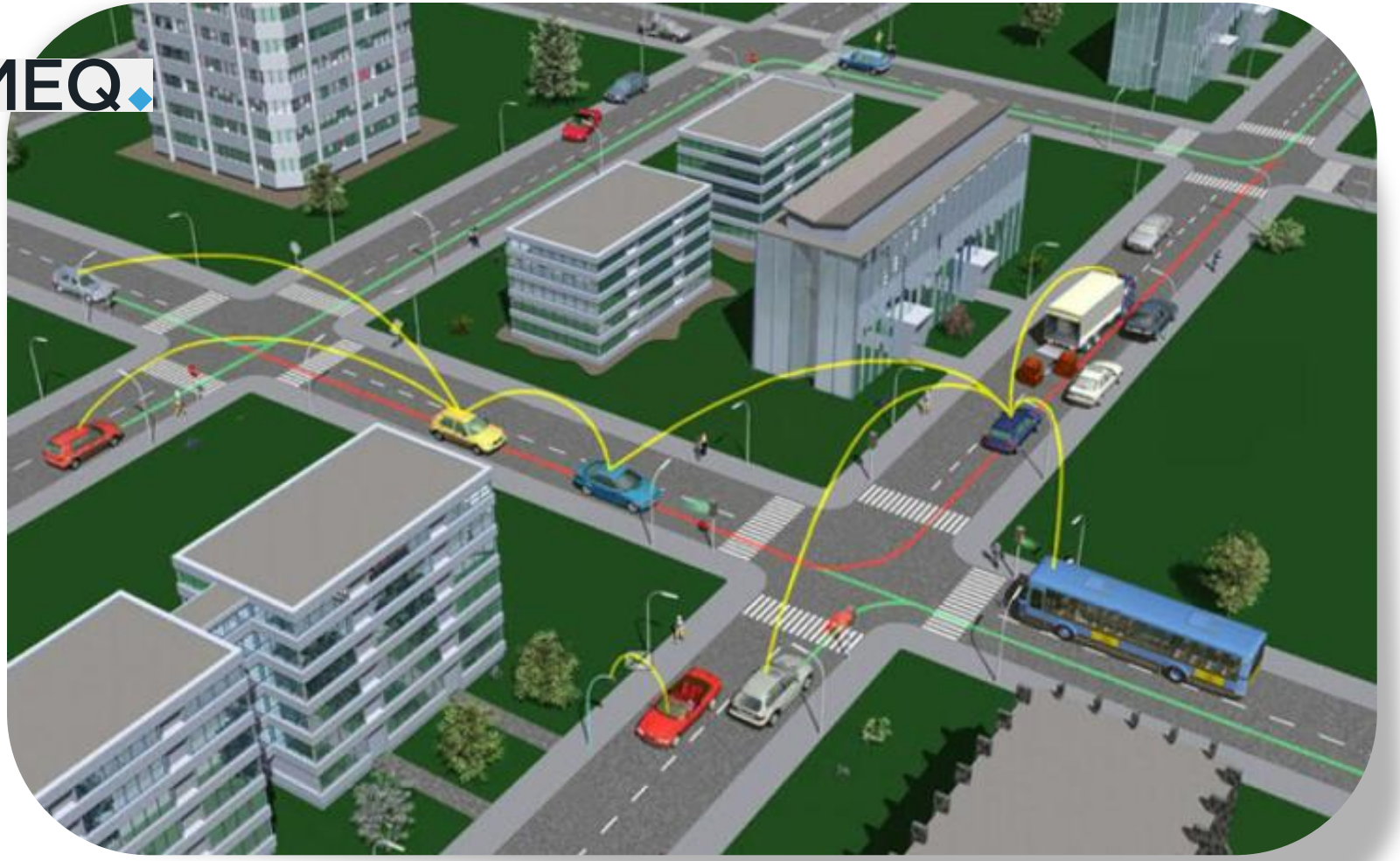
Integrated Water Treatment System



DANSE Test Cases

Automated Ground Transport

CARMEQ



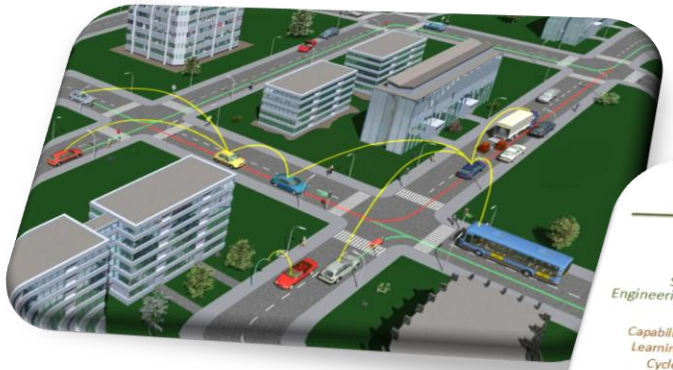
DANSE Test Cases
Air Traffic Management

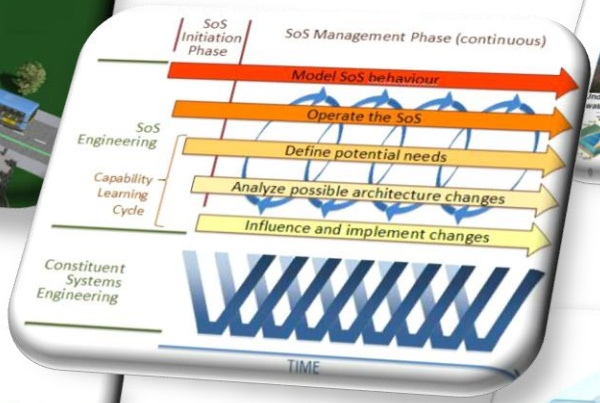
THALES

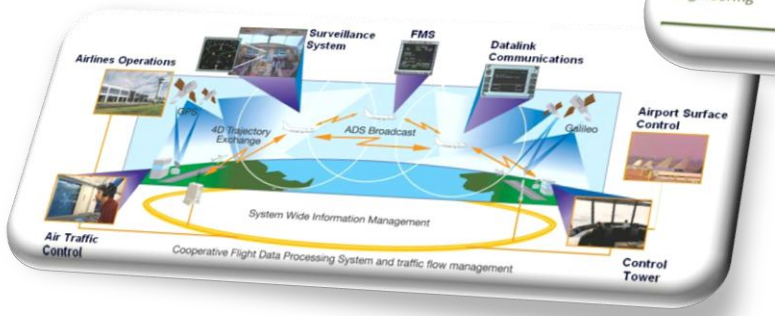
Honourcode, Inc.
Eric Honour
+1 (615) 614-1109
ehonour@hcode.com

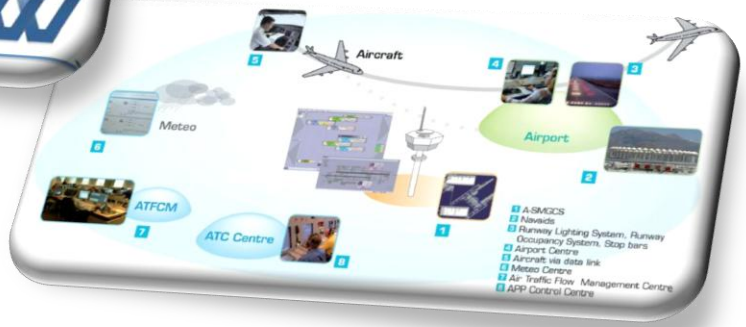
DANSE

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









SEVENTH FRAMEWORK PROGRAMME