Designing for Adaptability and evolution in System of systems Engineering

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

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

October 2014
Agenda

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

---

**Evolution**
- SoS
- sys model

---

**DANSE partners**
- business opportunities

---

**Honourcode, Inc.**

**Systems of Systems Concepts**
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

Differing Levels of “SoS-ness”
Emergency Response SoS

Slide by Tim Lochow, EADS
## Emergency Response SoS Dynamicity

**SoS operational timeline and dynamicity aspects**

<table>
<thead>
<tr>
<th>Modelling the SoS</th>
<th>Design Exploration Architecture Alternatives</th>
<th>Run Time Analysis &amp; Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Decades</strong></td>
<td>Population increase</td>
<td></td>
</tr>
<tr>
<td>Life Cycle</td>
<td>New fire, police and health care department stations are built or moved (More stations in order to serve smaller city areas)</td>
<td></td>
</tr>
<tr>
<td>Dynamics</td>
<td>New buildings, roads and crossroads are created</td>
<td></td>
</tr>
<tr>
<td><strong>Years</strong></td>
<td>More fire, police and health care department units are allocated</td>
<td></td>
</tr>
<tr>
<td>System Dynamics</td>
<td>New C4I command &amp; control organization &amp; communication system (e.g. introduction of LTE)</td>
<td></td>
</tr>
<tr>
<td><strong>Hours</strong></td>
<td>Improved Emergency response performance in terms of response time to emergency call and situational awareness</td>
<td></td>
</tr>
<tr>
<td><strong>Minutes</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Slide by Tim Lochow, EADS*
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)
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
What actions can an SoS manager/architect perform within the DANSE methodology?
## Solution Methods

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

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

Constituent Systems

MATLAB SIMULINK

MODELICA

OMG SYSTEMS MODELING LANGUAGE

GCSL Editor

Architecture Optimization Workbench

Architecture Generation

Architecture Patterns

Statistical Model Checking

Tool-Net SMC Platform

UPDM Rhapsody

Joint Simulation

Contract-based run-time verification

Contract analysis

Fault-based test generation

PLASMA

DESYRE
Example “Use Case” of Methodology

SoS Requirements Analysis → SoS Goals/Contracts

CS Modelling → SoS Modelling

SoS Architecture Model (UPDM/NAF/etc.) → Patterns

SoS Modelling → Architecture Optimization

SoS Modelling → Alternative Architecture Generation

Joint Simulation

Statistical Model Checking

Emergent Behaviour

Parametric Analysis

Formal Verification

Activities

Products
DANSE Tools

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

### Modeling

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

“Unified Profile for DoDAF and MoDAF,” also covers NAF

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-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

<table>
<thead>
<tr>
<th>View</th>
<th>Name</th>
<th>Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>OV-5a</td>
<td>Operational Activity Decomposition Tree</td>
<td>Structure of OV executable elements</td>
</tr>
<tr>
<td>OV-5b</td>
<td>Operational Activity Model</td>
<td>SysML Use Case, Activity forms</td>
</tr>
<tr>
<td>OV-6b</td>
<td>State Transition Description</td>
<td>SysML State diagrams</td>
</tr>
<tr>
<td>OV-6c</td>
<td>Event-Trace Description</td>
<td>SysML Sequence diagrams</td>
</tr>
<tr>
<td>SV-1</td>
<td>Systems Interface Description</td>
<td>Structure of SV executable elements</td>
</tr>
<tr>
<td>SV-4</td>
<td>Systems Functionality Description</td>
<td>SysML Use Case, Activity forms</td>
</tr>
<tr>
<td>SV-10b</td>
<td>Systems State Transition Description</td>
<td>SysML State diagrams</td>
</tr>
<tr>
<td>SV-10c</td>
<td>Systems Event-Trace Description</td>
<td>SysML Sequence diagrams</td>
</tr>
</tbody>
</table>
Constituent System Models

SoS View

authority

use

communicate

CS1

CS2

System View

Lift systems on the level of SoS

Com-Link

sendToDistrict(…)

requestStatus(…)

Abstraction Methods
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
- **Timing**: Focus on timing issues; other issues ignored
- **Statistical**: Match statistical behavior w/o details

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

Example GCSL statements:

0 - \( \text{Always}(true) \cdot \text{Whenever} \) 
\( \text{SoS.itsDistricts.fireArea} \cdot \text{sum}() > 0 \) occurs
\( \text{SoS.itsDistricts.fireArea} \cdot \text{sum}() = 0 \) occurs within [0, false]

1 - \( \text{Always}(true) \cdot \text{Mean} \) 
\( \text{SoS.itsDistricts.fireArea} \cdot \text{sum}() \cdot \text{duration/interval} < 0.01/100 \times \text{SoS.itsDistricts.area} \cdot \text{sum}() \)
# Solution Methods
## Architecture Exploration

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

- 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

Honourcode, Inc.
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
- Revised models created automatically by applying the rules
SoS contract
• Assumption
  ... (GCSL)
• Promise
  ... (GCSL)
Reachability of Future Architectures

- Contract Violation
- Low Goal Satisfaction
- OK

Centralized

Intermediate

Intermediate

Intermediate

Intermediate

Intermediate

Intermediate

Intermediate

Intermediate

Intermediate

Decentralized

Intermediate

Intermediate

Intermediate

Intermediate

Intermediate

Intermediate

Intermediate

Intermediate

Intermediate

Contract Violation

Low Goal Satisfaction

OK

Architectures Generation: Graph Grammar
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

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

4. Optimized architecture back annotated to SysML model
**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

- 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

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

The probability that a district does not stay under fire more than 3 hours shall be greater than 99.9%
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
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

SoS Requirements Analysis → SoS Goals/Contracts

CS Modelling → SoS Modelling

SoS Architecture Model (UPDM/NAF/etc.) → Patterns

SoS Modelling

Joint Simulation

Architecture Optimization → Alternative Architecture Generation

Statistical Model Checking

Emergent Behaviour → Parametric Analysis

Activities

Products

Formal Verification
Designing for Adaptability and evolution in System of systems Engineering

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

October 2014