Simplifying (and enriching) SysML to perform functional analysis and model instances

J.-L. Voirin, S. Bonnet, D. Exertier, V. Normand
INCOSE Symposium, July 20th, 2016
Thales: A Wide Spectrum of Complex Systems

**N°1 worldwide**

- Payloads for telecom satellites
- Air Traffic Management
- Sonars
- Security for interbank transactions

**N°2 worldwide**

- Rail signalling systems
- In-flight entertainment and connectivity
- Military tactical communications

**N°3 worldwide**

- Avionics
- Civil satellites
- Surface radars
Models, What For?

- **Answer questions**
  - About the system
    - What is it, how does it work, is the performance adequate, what happens if something breaks?
  - About the design
    - Is it complete, does it support required analyses, does it support impact analysis?

- **Ensure consistency**
  - Across different views, between upstream and downstream engineering, etc.

- **Generate artefacts**
  - Documentation (specification, architecture, interfaces)
  - Pieces of code, database schemas, configuration data, deployment data, etc.
Choose (and adapt) the right modeling solution for your objectives!
Arcadia and Capella

MODEL-BASED METHOD FOR ARCHITECTURAL DESIGN AND ITS SUPPORTING OPEN SOURCE MODELING WORKBENCH
Arcadia

Operational Need Analysis

System Need Analysis

Conceptual Solution

Finalized Solution
Capella: An Open Source Modeling Workbench Supporting Arcadia

Not a talk about Capella features, but….

- Methodological browser
- Semantic browser
- Computed graphical views
- Advanced diagram mgt.
- Validation & quick fixes
- Semantic delete
- Replicable elements
- Patterns
- HTML generation
- Transition to sub-systems
- Multi-viewpoint mgt.
Arcadia-Capella versus and SysML
NOT A PROFILE, NOT A DSL, BUT AN HYBRID APPROACH
Tooling a Model-Based Engineering Method

Existing language?

Yes

No

Customization of an existing language?

No

Yes

DSML

Customization of the tool?

No

Yes

Thales - 2016 All rights reserved
Back in the past (2003-2008)
Capella Core Concepts: The Wheel is Not Reinvented...

- FUNCTIONS, DATAFLOWS
- STRUCTURE
- INTERFACES
- BEHAVIOUR

July 20th, 2016
... Things Are Just Simpler ... when possible

Functions = Green

Components = Blue

Interfaces = Pink
Functional Analysis Workflows

Top-down
Functional Analysis Workflows
Functional Analysis Workflows

1. Bottom-Up

2. Top-Down

Bottom-Up
SysML Activity Diagrams vs Capella Functions

Rigid encapsulation and delegation mechanism, with (at least) three different kinds of « functions »

No delegation. When the design is complete, only leaf functions are supposed to have incoming/outgoing exchanges.
Functional Analysis with Capella
Functional Analysis with Capella

MODEL

VIEW

F1

F2

F21

F22

2
Graphical simplification: Ports on F21 and F22 do not actually « belong » to F21 and F22 but to their children functions.
Graphical simplification: Ports on F2 do not actually «belong» to F2 but to its children functions.
Graphical simplification: Ports on F1 and F2 do not actually « belong » to F1 and F2 but to their children functions.
Functional Analysis with Capella

Introduction of the « Category » concept
Graphical simplification based on the « Category » concept
Language and tooling work together to address practitioner’s engineering challenges and support different workflows.

Computed graphical simplifications are key to manage complexity.
How to deal with Sequence Flows (aka Control Flows)?

Example: Vehicle service checkup

Mr Jones

Hot engine scenario:
1. Change the oil
2. Wait for engine to be cold
3. Check coolant

Mr Smith

Cold engine scenario:
1. Check coolant
2. Warm engine up
3. Change the oil
Pure Control Flows: Contradictory with Dataflow Principles?

Example: vehicle service checkup

- 8:00 AM Mr Jones
  - Engine is hot
  - Take over the vehicle
  - Check & deliver the vehicle
  - Check & add coolant 1
  - Charge & deliver the vehicle 1
  - Change the oil 1

- 10:00 AM Mr Smith
  - Engine is cool
  - Take over the vehicle 1
  - Change the oil 1
  - Check & add coolant 1
  - Charge & deliver the vehicle 1

Issues

- Two antagonist dataflows: cannot coexist as is
- Do not express functional dependency
  - E.g. ‘Change the oil’ does not expect anything likely to come from ‘Check & add coolant’
Control Flows Preclude True Functional Analysis

- Focusing on what each function has to deliver and what it needs for so should prevail
  - This one is better from this point of view

- But it is not yet optimal
  - If engine is hot, no need to heat it first
Functional Dataflow Must Address All Use Cases and Contexts

- Checking all possible providers of an input leads to better analysis
  - This one is the most precise, leaving room for several use cases
Scenarios/functional Chains describe Contextual Behaviour

Mr Jones context

Applicable only inside one context (Capability, Mode, State...)
Express the « Use » of the dataflow inside this context
Focus on Instance-Driven Modeling
Most systems engineers think in terms of instances, not types!
Instance-Driven Modeling

**SysML**
- Blocks have Parts, typed by other Blocks
- Blocks can have the “PropertySpecificType” stereotype, emulating an instance-level modeling
- Activities have Partitions
- CallBehaviorActions belong to Partitions and invoke Activities
- Partitions represent either by Blocks or Parts
- Activities have ParameterNodes
- Actions have Pins
- Blocks have FlowPorts
- Blocks are related to each other via Associations
- Parts do not have their own FlowPort “instances”
- No diagram showing simultaneously Component and Activity/Actions

**Arcadia-Capella**
- Functions are allocated to Components
- By default, one Component ↔ one Part
- Functions and Components have Ports
- Any set of element can be part of a REC (record) or a RPL (replica)
- Content is synchronized between RPL and RECs
Instance-Driven Modeling

**SysML**
- Blocks have Parts, typed by other Blocks
- Blocks can have the "PropertySpecificType" stereotype, emulating an instance-level modeling
- Activities have Partitions
- CallBehavior:Actions belong to Partitions and
  Parts of Actions do not have their own "had/hasInstances"

**Arcadia-Capella**
- Functions are allocated to Components
- By default, one Component == one Part
- Functions and Components have Parts
- Any set of segment can be part of a RPL or a REC
- Content is synchronized between RPL and REC structures

Extremely rich (but complex) language

The same language concepts are used both for type and instance modeling
Trying with SysML Concepts

<table>
<thead>
<tr>
<th>Struggling with our “simple” needs</th>
</tr>
</thead>
<tbody>
<tr>
<td>• How do we specify “at type” level what Activities or Actions can be / are performed by a block? …</td>
</tr>
<tr>
<td>• … in order to constraint the Activities or Actions that can be in a Partition representing a Part by what has been defined on its typing Block?</td>
</tr>
<tr>
<td>• How can we model building block, assembly rules, and deployments that are consistent with each other?</td>
</tr>
<tr>
<td>- Including the Activities and Actions “instances” at each level?</td>
</tr>
<tr>
<td>- Including Port and Connection “instances” at each level?</td>
</tr>
</tbody>
</table>
Instance-Driven Modeling: The Capella Solution

The context and the model organization define whether an element is part of a Type or part of an Instance. Examples:

- Library (REC) vs projects (RPL)
- Assembly description (REC) vs deployment description (RPL)
- Building block (REC) vs assembly description (RPL)

An element can be simultaneously part of a REC and of a RPL
Instance-Driven Modeling: The Capella Solution

A « type » can be anything, including multi-root sets of elements
Instance-Driven Modeling: The Capella Solution

Support of multiple workflows
Visit us on Polarsys/Capella booth!

Capella website: http://www.polarsys.org/capella/
LinkedIn https://www.linkedin.com/company/capella-modelling-workbench
Twitter https://twitter.com/capella_arcadia
Arcadia forum: https://polarsys.org/forums/index.php/f/12/
Capella forum: https://polarsys.org/forums/index.php/f/13/
IFE model & doc.: http://www.polarsys.org/capella/start.html
www.thalesgroup.com

July 20th, 2016