Arcadia/Capella: Looking back at our implementation issues

CONSIDERATIONS FOR SYSML V2

Stéphane Bonnet
INCOSE WG, Torrance, Jan 28th, 2017
Tight coupling method/tool
A practitioner-driven journey started in Thales...

Practitioner-driven Capella development

Multi-domain Thales working groups on language and method

... now open source

http://www.polarsys.org/capella

Initial 3-year (French) collaborative project

Larger industry consortium currently being initiated
Back in the past (2003-2007)
Some drivers for an MBSE solution

- Be accessible to all practitioners (including non-architects)
- Reduce incidental complexity
- Help cope with design complexity
- Be flexible: support of multiple workflows
Insights (or, why Capella is not SysML...)

1. Functional analysis

2. Instance-driven modeling

3. Model exploitation

4. Decorrelate model/view
1. Functional analysis

2. Instance-driven modeling

3. Model exploitation

4. Decorrelate model/view
Functions in SysML vs Capella

1. **SysML**

   - F1, F121, F122, F21, F22 are OpaqueActions
   - F1, F2, F12 are CallBehaviorActions referencing Activities

2. **Capella**

   - Alternative representation 1
   - Alternative representation 2, etc.
“Natural representation”
Functional analysis workflows

Top-down

1

2
Functional analysis with Capella
1. Functional analysis

2. Instance-driven modeling

3. Model exploitation

4. Decorrelate model/view
Instance-Driven Modeling

A lot of systems engineers think first in term of instances
Instance-Driven Modeling

Building Blocks

Assembly description

Focused analyses

Deployments
Flexible Type/Instance Modeling
Flexible Type/Instance Modeling
Flexible Type/Instance Modeling
4 years ago: Explicit & implicit, semi-failure

- «Types» oriented representation, very common in UML / SysML
  - Parts specify the multiplicity and the role of the usage of one component in another
  - Avoids the multiplication of elements in the models
    - Describes the assembly of Components from an abstract point of view
    - Exactly the same as Class / Property relationships

- Limitations
  - If Component Properties are defined on C3, it is not possible to specify that the occurrences of C3 in C1 and in C2 have different default values
4 years ago: Explicit & implicit, semi-failure

- A dedicated SysML construction (PropertySpecificType) allows specifying specific information when needed (valuation of Component Properties for example)
  - Local specialization of the referenced Type
  - Still not equivalent to a real instance-level modeling
4 years ago: Explicit & implicit, semi-failure

- Allocation of two Functions to the Component C3

- On a xAB diagram,
  - Functions appear as many times as C3 appear, but all the green boxes are representing the same model elements (8 Functions boxes displayed but only 2 Functions in the model)
  - Exchanges are “multiplied” even though they are not meaningful

- In Arcadia, distinguishing the different occurrences / execution / instances of Functions is most of the time mandatory.

⇒ Explicit modeling is likely to be preferred.
4 years ago: Explicit & implicit, semi-failure

> When distinguishing the different executions / occurrences of Functions is necessary, modeling explicitly is required.

> The proposed solution is to exploit the local specialization mechanism

- Functions are duplicated & allocated to local specializing components
- When reusing a Component through a Part, local Functions are automatically created
Both implicit and explicit mode should be provided

- Be able to configure a default mode for part creation [implicit/explicit]
- When using implicit part, provide a way to make local specialization on demand (for example, when local valuation of a Property is needed)
- Additional request: The local specialization mechanism should be available for Classes as well
Component / Local Specialization mechanism CAN BE IMPLEMENTED WITH REC / RPL

Synchronization

Add element

Replicable element

Replica

Several kinds of compliance
1. Functional analysis

2. Instance-driven modeling

3. Model exploitation

4. Decorrelate model/view
Tooling and API Perspective

<< Give me all the Logical Components and their lists of Functions >>

Model extension and exploitation
Viewpoints, validation rules, queries, diff/merge, bridges, etc.

Domain / Method API

UML/SysML API

Domain or method meta-model

LC  LF

<< Give me all the Blocks with « LC » stereotype and their lists of Actions with « LF » stereotypes >>
1. Functional analysis
2. Instance-driven modeling
3. Model exploitation
4. Decorrelate model/view