Short Introduction about Systems Engineering and SysML

Benoit Combemale

DiverSE team (IRISA & Inria)
University of Rennes 1
benoit.combemale@irisa.fr

Inspired from the OMG specification SysML v1.2, from the OMG/INCOSE tutorial, from the Prof. J.-M. Bruel lecture, and the G. Finance’s article (Object Direct).


Materials available on: http://combemale.fr/
System Complexity
System Complexity

System Complexity

But also...

lines of code
### System Complexity

<table>
<thead>
<tr>
<th>Language</th>
<th>Code Lines</th>
<th>Comment Lines</th>
<th>Comment Ratio</th>
<th>Blank Lines</th>
<th>Total Lines</th>
<th>Total Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>2,300,710</td>
<td>475,978</td>
<td>17.2%</td>
<td>452,773</td>
<td>3,332,461</td>
<td>34.4%</td>
</tr>
<tr>
<td>C++</td>
<td>1,206,025</td>
<td>250,128</td>
<td>17.2%</td>
<td>252,971</td>
<td>1,709,124</td>
<td>16.3%</td>
</tr>
<tr>
<td>Java</td>
<td>743,003</td>
<td>698,939</td>
<td>46.5%</td>
<td>179,667</td>
<td>1,622,829</td>
<td>17.4%</td>
</tr>
<tr>
<td>Ada</td>
<td>729,322</td>
<td>335,302</td>
<td>31.5%</td>
<td>252,866</td>
<td>1,317,510</td>
<td>14.1%</td>
</tr>
<tr>
<td>Autoconf</td>
<td>450,574</td>
<td>755</td>
<td>0.2%</td>
<td>71,979</td>
<td>523,309</td>
<td>5.6%</td>
</tr>
<tr>
<td>HTML</td>
<td>214,572</td>
<td>6,279</td>
<td>2.8%</td>
<td>43,661</td>
<td>264,512</td>
<td>2.8%</td>
</tr>
<tr>
<td>Fortran (Fixed-format)</td>
<td>113,138</td>
<td>2,323</td>
<td>2.0%</td>
<td>15,909</td>
<td>131,373</td>
<td>1.4%</td>
</tr>
<tr>
<td>Make</td>
<td>112,507</td>
<td>3,917</td>
<td>3.4%</td>
<td>14,123</td>
<td>130,547</td>
<td>1.4%</td>
</tr>
<tr>
<td>Go</td>
<td>88,921</td>
<td>11,083</td>
<td>14.2%</td>
<td>4,004</td>
<td>92,908</td>
<td>0.0%</td>
</tr>
<tr>
<td>Assembly</td>
<td>51,774</td>
<td>13,375</td>
<td>20.5%</td>
<td>10,980</td>
<td>75,229</td>
<td>0.8%</td>
</tr>
<tr>
<td>XML</td>
<td>49,875</td>
<td>675</td>
<td>1.3%</td>
<td>6,002</td>
<td>50,612</td>
<td>0.0%</td>
</tr>
<tr>
<td>Objective-C</td>
<td>28,137</td>
<td>5,215</td>
<td>15.6%</td>
<td>8,279</td>
<td>41,631</td>
<td>0.0%</td>
</tr>
<tr>
<td>shell script</td>
<td>19,657</td>
<td>5,823</td>
<td>22.9%</td>
<td>4,417</td>
<td>29,897</td>
<td>0.0%</td>
</tr>
<tr>
<td>Fortran (Free-format)</td>
<td>17,058</td>
<td>3,305</td>
<td>16.2%</td>
<td>1,666</td>
<td>22,059</td>
<td>0.2%</td>
</tr>
<tr>
<td>Perl</td>
<td>15,549</td>
<td>3,869</td>
<td>18.9%</td>
<td>2,463</td>
<td>22,881</td>
<td>0.2%</td>
</tr>
<tr>
<td>TeX/LaTeX</td>
<td>12,823</td>
<td>6,358</td>
<td>33.1%</td>
<td>1,699</td>
<td>20,292</td>
<td>0.2%</td>
</tr>
<tr>
<td>Scheme</td>
<td>11,023</td>
<td>1,010</td>
<td>8.4%</td>
<td>1,205</td>
<td>13,238</td>
<td>0.1%</td>
</tr>
<tr>
<td>Automake</td>
<td>10,775</td>
<td>1,210</td>
<td>10.1%</td>
<td>1,626</td>
<td>13,611</td>
<td>0.1%</td>
</tr>
<tr>
<td>Modula-2</td>
<td>4,325</td>
<td>983</td>
<td>16.5%</td>
<td>826</td>
<td>6,135</td>
<td>0.1%</td>
</tr>
<tr>
<td>Objective Caml</td>
<td>2,930</td>
<td>579</td>
<td>16.5%</td>
<td>369</td>
<td>3,897</td>
<td>0.0%</td>
</tr>
<tr>
<td>XSL Transformation</td>
<td>2,896</td>
<td>450</td>
<td>13.4%</td>
<td>576</td>
<td>3,922</td>
<td>0.0%</td>
</tr>
<tr>
<td>AWK</td>
<td>2,318</td>
<td>569</td>
<td>19.7%</td>
<td>376</td>
<td>3,263</td>
<td>0.0%</td>
</tr>
<tr>
<td>CSS</td>
<td>2,049</td>
<td>171</td>
<td>7.7%</td>
<td>453</td>
<td>2,673</td>
<td>0.0%</td>
</tr>
<tr>
<td>Python</td>
<td>1,735</td>
<td>410</td>
<td>19.1%</td>
<td>404</td>
<td>2,549</td>
<td>0.0%</td>
</tr>
<tr>
<td>Pascal</td>
<td>1,044</td>
<td>141</td>
<td>11.9%</td>
<td>218</td>
<td>1,403</td>
<td>0.0%</td>
</tr>
<tr>
<td>C#</td>
<td>879</td>
<td>508</td>
<td>36.5%</td>
<td>220</td>
<td>1,615</td>
<td>0.0%</td>
</tr>
<tr>
<td>DCL</td>
<td>693</td>
<td>154</td>
<td>16.1%</td>
<td>15</td>
<td>867</td>
<td>0.0%</td>
</tr>
<tr>
<td>JavaScript</td>
<td>655</td>
<td>404</td>
<td>36.1%</td>
<td>144</td>
<td>1,203</td>
<td>0.0%</td>
</tr>
<tr>
<td>Tel</td>
<td>392</td>
<td>113</td>
<td>22.4%</td>
<td>72</td>
<td>487</td>
<td>0.0%</td>
</tr>
<tr>
<td>Haskell</td>
<td>154</td>
<td>0</td>
<td>0.0%</td>
<td>17</td>
<td>154</td>
<td>0.0%</td>
</tr>
<tr>
<td>CMake</td>
<td>134</td>
<td>31</td>
<td>18.8%</td>
<td>25</td>
<td>190</td>
<td>0.0%</td>
</tr>
<tr>
<td>Matlab</td>
<td>57</td>
<td>0</td>
<td>0.0%</td>
<td>8</td>
<td>57</td>
<td>0.0%</td>
</tr>
<tr>
<td>DOS batch script</td>
<td>4</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
<td>4</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

**Totals**: 6,174,724 | 1,832,058 | 1,330,033 | 9,337,085

- Interoperability

System Complexity

- Reusability
- Durability

=> Time To Market!

- Variability
System Complexity

- Embedded
- Critical
- Real time

**Phaeton**
- 61 networked ECUs
- 3 bus systems + optical bus + sub busses
- 2500 signals in 250 CAN-messages
- more than 50 MByte memory
- more than 2000 individual wires
- more than 3800m cables
System Complexity

Flight project teams are large

Socio-technical coordination

Flight Project Impact: Reduction of Overhead

- 100 Concurrent users
- 230+ Documents and decision gates deliverables including
- 445,000 Connections between elements

Aerospace Industry Impact: Enterprise Scalability

- 1,000 Concurrent users
- 50 Programs
- 100 Concurrent users
- 50 Projects
System Complexity

- Autonomic Computing
- Cloud Computing
- SaaS, IoS, IoT
- System of Systems
System Complexity
Failures in Civil Engineering!
Systems Complexity: Some Dimensions

Interoperable
Real Time
Large-Scale
Secure
Parameterized
Interoperable
Pervasive
Reusable
Distributed
Durable
Embedded
Variable
Adaptable
Dynamic
Autonomous
Secure
Usable!
Heterogeneous

Outline

• From **Software** Engineering to **Systems** Engineering

• **SysML**: Overview

• **SysML** Structure Diagrams

• **SysML** Behavioral Diagrams

• **SysML Extensions**: Requirement Diagram & Allocation

• Conclusion
Systems Engineering (SE) ...

- ... is an approach and discipline to deal with complex systems realized through software and hardware solutions.

- ... relies on modelling and simulation methods to validate requirements or to evaluate the system.

- ... applies to the following areas and industries: embedded systems (e.g. audio and video encoding/decoding, set top box, home automation, smart building, smart city, etc.), transport (automotive, rail, avionics, etc.), factories, military, telecom, healthcare, energy, etc.
Systems Engineering (SE) ...

- focuses on:
  - defining customer needs and required functionality early in the development cycle
  - documenting requirements
  - design synthesis and system validation
- considers the complete problem:
  - Operations, Cost & Schedule, Performance, Training & Support, Test, Disposal, Manufacturing...
- integrates all the disciplines and specialty groups that proceeds from concept to production to operation
- considers both the business and the technical needs
Systems Engineering (SE) ...

• The International Council on Systems Engineering

• **Mission:** Share, promote and advance the best of SE

• **Vision:** The world's authority on Systems Engineering

• **Goals:**
  - To provide a focal point for dissemination of SE knowledge
  - To promote collaboration in SE practice, education, and research
  - To assure the establishment of competitive, scaleable professional standards in the practice of SE
  - To improve the professional status of all persons engaged in the practice of SE
  - To encourage governmental and industrial support for research and educational

• *Cf. [http://www.incose.org/](http://www.incose.org/)*
The SIMILAR Process (INCOSE)

- **State the problem**
- **Investigate alternatives**
- **Model the system**
- **Integrate**
- **Launch the system**
- **Assess performance**
- **Re-evaluate**

Outline

• From Software Engineering to Systems Engineering

• SysML: Overview

• SysML Structure Diagrams

• SysML Behavioral Diagrams

• SysML Extensions: Requirement Diagram & Allocation

• Conclusion
The advent of SysML...

- 01/1997 : UML v1.0
- 2001 : INCOSE & OMG form the Systems Engineering Domains Special Interest Group (SE DSIG)
- 03/2003 : UML for Systems Engineering RFP
- 06/2003 : MDA Guide v1.0.1
- 01/2005 : UML v1.4.2 (ISO/IEC 19501)
- 07/2005 : UML v2.0
- 07/2006 : SysML is officially adopted by the OMG
- 09/2007 : SysML v1.0
- 11/2008 : SysML v1.1
- 08/2011 : UML v2.4.1
- 05/2017 : SysML v1.5 (current version)
- 12/2017 : UML v2.5.1 (current version)
- 03/2019 : SysML v1.6 beta
- *Towards SysML v2!*
SysML: Who is behind?

• **Industry**

• **Tool vendors**

• **Other organisations**
  - *AP-233, INCOSE, Georgia Institute of Technology, AFIS, …*
SysML: a modeling language for SE

• Standard modeling language for SE to analyze, specify, design, and verify complex systems

• Intended to
  ▪ enhance systems quality
  ▪ improve the ability to exchange systems engineering information amongst tools
  ▪ help bridge the semantic gap between systems, software, and other engineering disciplines
SysML Overview

• is based on UML (v2.x)

• involves modeling blocks instead of modeling classes

• provides a vocabulary that’s more suitable for SE
SysML Overview

- **SysML**: the OMG Systems Modeling Language
  - Systems Engineering
- **UML**: the OMG Software Modeling Language
  - Software Engineering

[Diagram showing the relationship between UML and SysML]
UML: 13 diagrams
SysML: $13 - 6 + 2 = 9$ diagrams
SysML diagrams

• **Structure diagrams**
  - The Block Definition Diagram (BDD), replacing the UML2 class diagram
  - The Internal Block Diagram (IBD), replacing the UML2 composite structure diagram
  - The Parametric Diagram, a SysML extension to analyse critical system parameters
  - The Package Diagram remains unchanged

• **Dynamic diagrams**
  - The activity diagram has been slightly modified in SysML
  - The sequence, state chart, and use case diagrams remain unchanged

• **The requirements diagrams is a SysML extension**
SysML diagrams

The Four Pillars of UML

Class diagram

Sequence diagram

UseCase diagram

Implementation diagram
The Four Pillars of SysML

1. Structure

2. Behavior

3. Requirements

4. Parametrics

Note that the Package and Use Case diagrams are not shown in this example, but are respectively part of the structure and behavior pillars

SysML diagram frames

• Each SysML diag. represents a model element
• Each SysML diag. must have a Diagram Frame
• Diagram context is indicated in the header:
  ▪ Diagram kind (req, act, bdd, ibd, sd, etc.)
  ▪ Model element type (package, block, activity, etc.)
  ▪ Model element name
  ▪ User defined diagram name or view name
• A separate diagram description block is used to indicate if the diagram is complete, or has elements elided
SysML diagram frames (e.g.)

![Diagram showing header, content, diag. type, and diag. name labels.]

Outline

• From **Software** Engineering to **Systems** Engineering

• **SysML**: Overview

  • **SysML Structure Diagrams**

  • **SysML Behavioral Diagrams**

  • **SysML Extensions**: Requirement Diagram & Allocation

• Conclusion
SysML Structure Diagrams

- Package Diagram
- Block Definition Diagram
- Internal Block Definition Diagram
- Parametric Diagram
Package Diagram (pkg)

• ~Same as UML

• To organize the model
  ▪ Groups model elements into a name space
  ▪ Often represented in tool browser
  ▪ Supports model configuration management (check-in/out)

• Model can be organized in multiple ways:
  ▪ System hierarchy (e.g., enterprise, system, component)
  ▪ Diagram kind (e.g., requirements, use cases, behavior)
  ▪ Use viewpoints to augment model organization

• Value Types: reusable types for properties or attributes in the model (new SysML extension)
Package Diagram (pkg)

**By Diagram Type**
- SampleModel [by diagram type]
  - Use Cases
  - Requirements
  - Behavior
  - Structure
  - EngrAnalysis

**By Hierarchy**
- SampleModel [by level]
  - Enterprise
  - System
  - Logical Design
  - Physical Design
  - Verification

**By IPT**
- SampleModel [by IPT]
  - Architecture Team
  - Requirements Team
  - IPT A
  - IPT B
  - IPT C

Copyright © 2006-2008 by Object Management Group.
OMG Distiller Example (pkg)

(pkg Distiller [model organization])

- Distiller Requirements
- Distiller Use Cases
- Distiller Behavior
- Distiller Structure

(pkg Value Types [value types for distiller])

- <<ValueTypes>> Real
- <<ValueTypes>> °C
  - dimension = temperature
  - unit = degrees celsius
- <<ValueTypes>> N/m²
  - dimension = pressure
  - unit = newtons per square meter
- <<ValueTypes>> gm/sec
  - dimension = mass flow rate
  - unit = grams per second
- <<ValueTypes>> cal/sec
  - dimension = heat flow rate
  - unit = calories per second
- <<ValueTypes>> cal/(gm°C)
  - dimension = specific heat
  - unit = calories per gram degree celsius
- <<ValueTypes>> cal/gm
  - dimension = latent heat
  - unit = calories per gram

Copyright © 2006-2008 by Object Management Group.
Block Definition Diagram (bdd)

- The BDD provides a black box representation of a system block alongside the hierarchy of its composite blocks.
- The BDD can include blocks of any type including software, hardware, etc.
- A block
  - provides a unifying concept to describe the structure of an element or system
  - encompasses software, hardware, data, processes, personnel, and facilities.
  - is shown as a UML class, stereotyped « block ».
SysML Block

- **Compartments**
  - Properties
  - Operations
  - Constraints
  - Allocations
  - Requirements

- **User defined!**

Copyright © 2006-2008 by Object Management Group.
OMG Distiller Example (bdd)

(from [Finance10])
Internal Block Diagram (ibd)

- Provides the white box or internal view of a system block
- Usually instantiated from the BDD to represent the final assembly
- Composite blocks from the BDD are instantiated on the IBD as parts
- Parts are assembled through connectors, linking them directly or via their ports (standard and/or flow ports)
- Redefines the UML2 composite structure diagram with blocks and flow ports.
Block Definition vs. Usage

**Definition:**
- Block is a definition/type
- Capture properties, etc.
- Reused in multiple contexts

**Usage:**
- Part is the usage of a block in the context of a composing block
- Also known as a role
Internal Block Diagram (ibd)

Enclosing Block

Connector

Item Flow

Port

Part

SysML Ports

- Specifies interaction points on blocks and parts
  - Integrates behavior with structure

• Standard (UML) Port:
  - Specifies a set of required or provided operations and/or signals
  - Typed by a UML interface

• Flow Port:
  - Specifies what can flow in or out of block/part
  - Typed by a block, value type, or flow specification
  - Atomic, non-atomic, and conjugate variations
SysML Ports: delegation

- to preserve **encapsulation** of block
- interactions at outer ports are **delegated** to ports of child parts
- ports must **match**
  - same kind, type, direction, etc.
- connectors can **cross boundaries** without requiring ports at each level of nested hierarchy
OMG Distiller Example (ibd)

(from [Finance10])
Parametric Diagram (par)

- To express constraints between value properties
  - equations
  - support for engineering analysis (e.g., performance)
  - identification of critical performance properties
- Constraint block captures equations
  - Expression language can be formal (e.g., MathML, OCL)
  - Computational engine is not provided by SysML
- Parametric diagram
  - usage of the constraints in an analysis context
OMG Distiller Example (par)
Outline

• From **Software** Engineering to **Systems** Engineering

• **SysML:** Overview

• **SysML Structure Diagrams**

• **SysML Behavioral Diagrams**

• **SysML Extensions:** Requirement Diagram & Allocation

• Conclusion
SysML Behavioral Diagrams

- Activity Diagram
- Sequence Diagram
- State Machine Diagram
- Use Case Diagram
Activity Diagram (act)

- to specify
  - controlled sequence of actions
  - the flow of inputs/outputs
  - control, including sequence and conditions for coordinate activities

- Swimlanes
  - to show responsibility of the activity
Activity Diagram (act)

• **Improvements from UML:**
  - *continuous* or discrete flow
  - *control* operators
    • to start/stop other actions
  - *Overwrite* and *NoBuffer* ports
    • for continuous flows
  - *probabilities* on transitions or parameter
Routing Flow

- **Initial Node** – On execution of parent control token placed on outgoing control flows

- **Activity Final Node** – Receipt of a control token terminates parent

- **Flow Final Node** – Sink for control tokens

- **Fork Node** – Duplicates input (control or object) tokens from its input flow onto all outgoing flows

- **Join Node** – Waits for an input (control or object) token on all input flows and then places them all on the outgoing flow

- **Decision Node** – Waits for an input (control or object) token on its input flow and places it on one outgoing flow based on guards

- **Merge Node** – Waits for an input (control or object) token on any input flows and then places it on the outgoing flow

Guard expressions can be applied on all flows
Activity Diagram (act)
Actions Process Flow of Control and Data

• Two types of flow:
  ▪ Object/Data and Control

• Unit of flow is called a «token» (consumed & produced by actions)
OMG Distiller Example (act)

Actions (Functions)  Control (Sequence)  Things that flow (ObjectNodes)
Interaction Diagrams (sdm, sd & uc)

State Machine, Sequence, and Use Case Diagrams:

Like in UML!
OMG Distiller Example (sd & uc)
OMG Distiller Example (sdm)
Outline

• From **Software** Engineering to **Systems** Engineering

• **SysML:** Overview

• **SysML Structure Diagrams**

• **SysML Behavioral Diagrams**

• **SysML Extensions: Requirement Diagram & Allocation**

• **Conclusion**
SysML Requirement Diagram

(SysML extension)
SysML Requirement Diagram

- `<<requirement>>` allows to represent a text based requirement and their dependencies
  - Includes one identifier id and some textual properties
  - Can add user defined properties
  - Can add user defined requirement categories

- Requirements can be
  - decomposed
  - specialized

- Requirement relationships

- `<<Problem>>` and `<<Rationale>>`:
  - can be attached to any model Element to capture *Issues* and *Decisions*
OMG Distiller Example (req)

Original Statement:
- Heat dirty water and condense steam are performed by Counter Flow Heat Exchanger
- Boil dirty water is performed by a Boiler
- Drain residue is performed by a Drain

Water has the following properties: vol = 1 litre, density = 1 g/ml, temp = 20 °C, specific heat = 1 cal/g °C, heat of vaporization = 640 cal/g

**Requirement:**
- The system shall purify water by boiling it

**Rationale:**
The requirement for a boiling function and a boiler implies that the water must be purified by distillation.

(from [Finance10])
Allocations

• Term from the systems engineers’ vocabulary
• General relationship between two elements of the model

• Different kinds of allocation:
  ▪ Functionality - component
  ▪ Logical component – physical component
  ▪ Software – hardware

• Explicit allocation of activities to structure via swim lanes (i.e., activity partitions)

• Usable under graphical or tabular representation

• Enables consistency in the model (e.g., between dynamic model elements and static model elements).
Allocations Representation

Allocate Relationship

Explicit Allocation of Action to Part Property

Compartment Notation

*Read as follows: “part name has constraints that are allocated to/from an <<element type>> Element Name”*

Callout Notation

Stereotypes & Model Libraries

• Mechanisms for further customizing SysML Profiles represent extensions to the language
  ▪ Stereotypes extend meta-classes with properties and constraints
    • Stereotype properties capture metadata about the model element
  ▪ Profile is applied to user model
  ▪ Profile can also restrict the subset of the meta-model used when the profile is applied

• Model Libraries represent reusable libraries of model elements
Stereotypes

Defining the Stereotype

Applying the Stereotype
Model Libraries
Outline

• From **Software** Engineering to **Systems** Engineering

• **SysML: Overview**

• **SysML Structure Diagrams**

• **SysML Behavioral Diagrams**

• **SysML Extensions: Requirement Diagram & Allocation**

• **Conclusion**
Cross Connecting Model Elements

1. Structure

2. Behavior

3. Requirements

4. Parametrics
Towards SysML v2

• A standalone, formal grounded, layered, and extensible language
• Come first with a textual syntax, and a graphical one as an alternative

• Current proposition: https://drive.google.com/drive/mobile/folders/1L0E3RwO9ch3Ta5Ye4EIdmn15yRPgwS4B

• Complete tutorial on demand
Conclusion

• **SysML is:**
  - a specific language for complex systems
  - strongly UML-Based
  - focusing on specification, analysis, design and verification

• **SysML is not:**
  - a method or a tool
  - just a UML profile
  - sufficient in itself
## Typical Integrated Tool Environment

<table>
<thead>
<tr>
<th>CM/DM</th>
<th>Product Data Management</th>
<th>Requirements Management</th>
<th>Verification &amp; Validation</th>
<th>Project Management</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SoS/ DoDAF / Business Process Modeling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>System Modeling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SysML</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Modeling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>UML 2.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hardware Modeling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VHDL, CAD, ..</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Simulation &amp; Visualization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Engineering Analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tools

- Artisan (Studio)
- EmbeddedPlus (SysML Toolkit)
- No Magic (Magic Draw / Cameo)
- Sparx Systems (Enterprise Architect)
- IBM (Tau and Rhapsody)

- Capella: https://www.eclipse.org/capella/
- Papyrus: https://www.eclipse.org/papyrus/
- Visio SysML template
Papyrus (SysML)

Official website: https://www.eclipse.org/papyrus/
Capella

- An industrial, open-source, systems engineering workbench
  - With its own formalism (~aligned with SysML)
  - With its own method (namely, Arcadia)

- Official website: https://www.eclipse.org/capella/
- The Arcadia method: https://www.eclipse.org/capella/arcadia.html

- Differences with SysML: https://www.eclipse.org/capella/arcadia_capella_sysml_tool.html
Arcadia

The Arcadia method: https://www.eclipse.org/capella/arcadia.html
References and links

• **Books:**

• **The Official OMG SysML site:**
  - http://www.omgsysml.org

• **INCOSE, International Council on Systems**
  - http://www.incose.org/

• **AFIS, Association Française d’Ingénierie Système**
  - http://www.afis.fr/

• **Association SysML France**
  - http://sysmlfrance.blogspot.com/

• **Misc:**