SysML tooling for design and engineering of System of Systems

Prof. Jerker Delsing

Luleå University of Technology

Sweden

ARROWHEAD TOOLS

What is SysML

The **Systems Modeling Language** (**SysML**)[1] is a general-purpose modeling language for systems engineering applications. It supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems.

SysML was originally developed by an open source specification project, and includes an open source license for distribution and use.[2] SysML is defined as an extension of a subset of the Unified Modeling Language (UML) using UML's profile mechanism. The language's extensions were designed to support systems engineering activities.

ARROWHEAD TOOLS

SysML Tools

MagicDraw - Cameo, commercial

SysML v1.6

Extensive graphical system modelling tool based on SysML

Papyrus - Eclipse Open source

SysML 1.6 and upcoming v2.0

ARROWHEAD TOOLS

System of Systems modelling

- Based on
  - Service Oriented Architecture
  - Micro-system producing and consuming micro-services

ARROWHEAD TOOLS

SOA/microsystem characteristics

ARROWHEAD TOOLS

SOA/microsystem characteristics

- Look up
  - Requires a service registry

ARROWHEAD TOOLS

SOA/microsystem characteristics

- Look up
  - Requires a service registry
- Late binding
  - Requires orchestration capability

ARROWHEAD TOOLS

SOA/microsystem characteristics

- Look up
  - Requires a service registry
- Late binding
  - Requires orchestration capability
- Loose coupling
  - Autonomous exchange of services, push or pull based

ARROWHEAD TOOLS

SOA/microsystem characteristics

- Look up
  - Requires a service registry
- Late binding
  - Requires orchestration capability
- Loose coupling
  - Autonomous exchange of services, push or pull based
- A micro system performs its function independently

ARROWHEAD TOOLS

SOA/microsystem characteristics

- Look up
  - Requires a service registry
- Late binding
  - Requires orchestration capability
- Loose coupling
  - Autonomous exchange of services, push or pull based
- A micro system performs its function independently
- A micro system can
  - be stateful and is then responsible for and stores its own state
  - be stateless

ARROWHEAD TOOLS

SOA/microsystem characteristics

- Look up
  - Requires a service registry
- Late binding
  - Requires orchestration capability
- Loose coupling
  - Autonomous exchange of services, push or pull based
- A micro system performs its function independently
- A micro system can
  - be stateful and is then responsible for and stores its own state
  - be stateless
- A micro system produces and/or consumes one or several services

ARROWHEAD TOOLS

SoS characteristics

Operational independence/autonomy of the elements. The constituent systems can operate independently in a meaningful way, and are useful in their own right.

Belonging. The autonomous constituent systems choose to belong to the SoS, and they do that because they see a value for themselves to give up some of the autonomy in order to get benefits from doing so.

Connectivity. To let the constituent systems interact, they must be connected, and unless they provide sufficiently generic interfaces, they need to be modified to provide such interoperability. Connectivity in an SoS is thus dynamic, with interfaces and links forming and vanishing as the need arises.

Diversity - heterogeneity. Whereas many other systems strive to minimize diversity to simplify the system, an increased diversity in an SoS gives it the ability to better deal with unforeseen situations during its life cycle.

Managerial independence of the elements. The constituent systems not only can operate independently, but they do operate independently even while being part of the SoS. They are acquired separately.

Evolutionary development. The SoS does not appear fully formed, and functions and purposes are added based on experience.

Emergent behavior. The principle purposes of the SoS are fulfilled by behaviors that cannot be localized to any individual constituent system. In an SoS, the emergent behavior is not restricted to what can be foreseen. Instead, it should have the capability to early detect and eliminate bad behavior that emerges.

Geographical distribution. The constituent systems only exchange information and not substantial quantities of mass or energy.

Secure and safe. Malicious behaviors in a SoS and its constituent systems need to be detected and mitigated to ensure information, system and SoS integrity.

Modelling of System of Systems, SoS

Based on Eclipse Arrowhead

A SOA/microsystem framework for creating automation and digitalisation solutions based on SoS

ARROWHEAD TOOLS

SysML modelling basics

Requirement diagram/table

Use case diagrams

Activity diagram

Block definition diagrams

Internal block diagrams

Parametric diagram

State machine diagram

Sequence diagrams

…

…

ARROWHEAD TOOLS

SOA support

Profile

- Based on Eclipse Arrowhead
- Intend to support several engineering phases for a solution
  - Requirement
  - Design conceptual, black box,
  - Design of implementation, white box,
  - Procurement & Engineering
  - Deployment

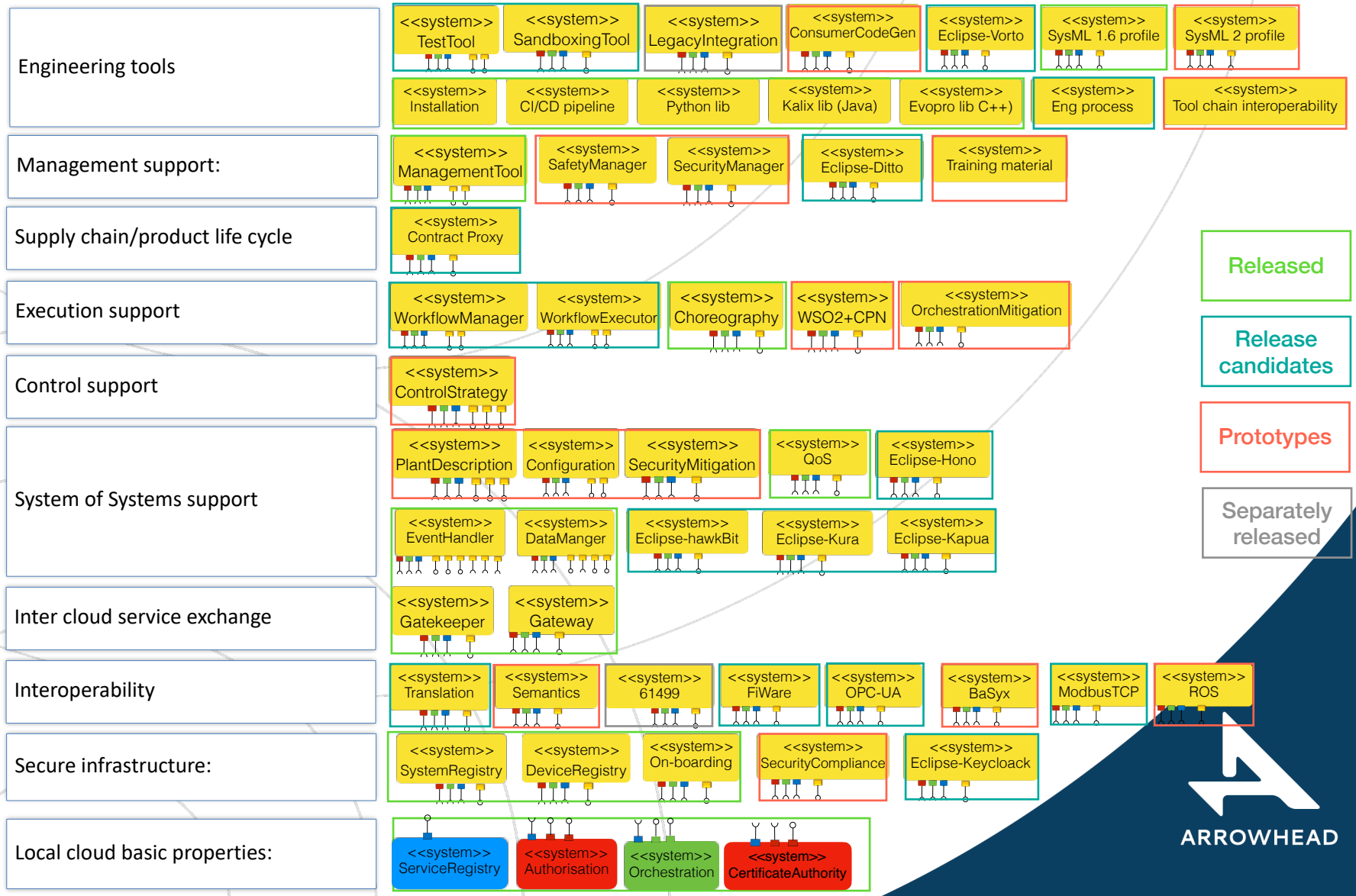
ARROWHEAD TOOLS

SOA support, cont

- Library
  - Eclipse Arrowhead core systems
  - Templates for
    - Local clouds
    - System of Local clouds
    - Generic application systems
    - Devices
    - Network

ARROWHEAD TOOLS

Arrowhead v4.3.0



Arrowhead v4.3.0

Engineering tools

Management support:

Supply chain/product life cycle

Execution support

Control support

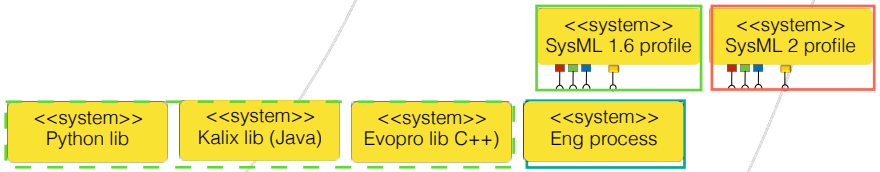
System of Systems support

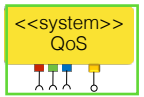
Inter cloud service exchange

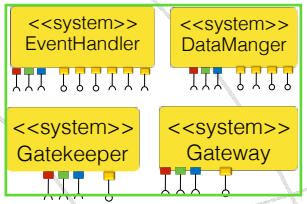
Interoperability

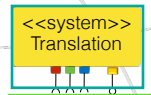
Secure infrastructure:

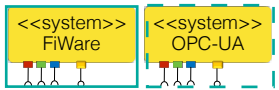
Local cloud basic properties:

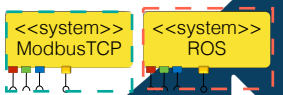


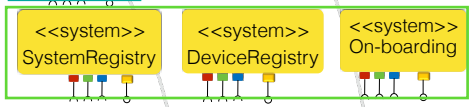


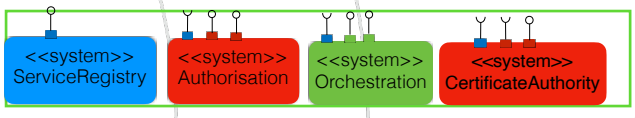












Released

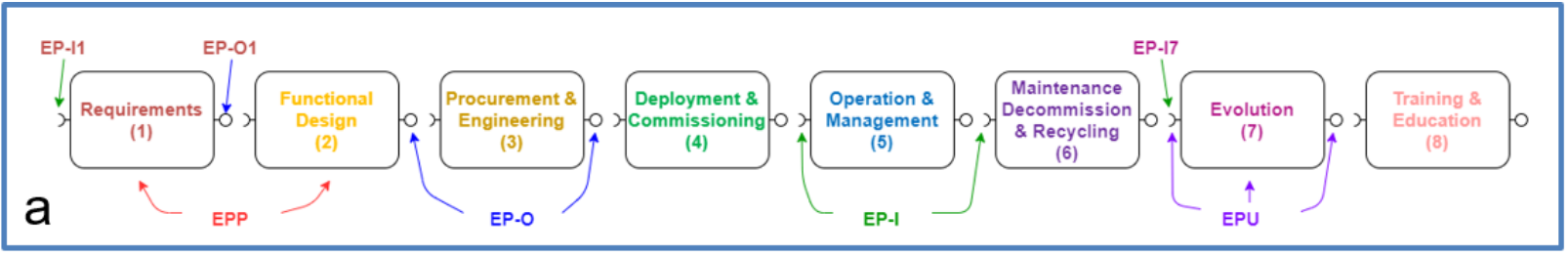
Release candidates

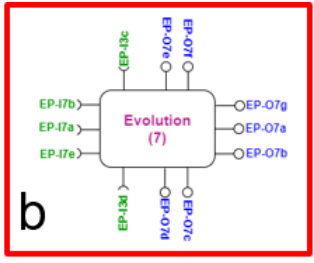
Prototypes

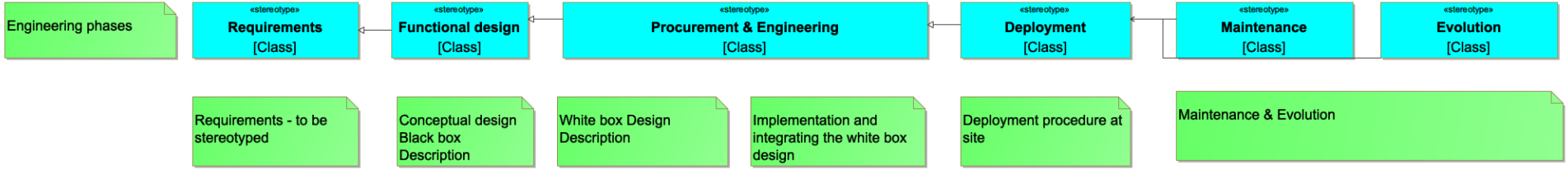
Separately released

ARROWHEAD

Engineering process







ARROWHEAD TOOLS

Eclipse Arrowhead engineering

Engineering process IEC 81346

Requirements — Functional design — Procurement & engineering — Deployment & commissioning

SoS Requirements
- Functional,
- non-functional,
- security,
- commissioning,
- operations,
- management,
- maintenance,
- evolution

SoLCD

Functional design - black box
- Plant architecture and design,
- Functionality design
- Security design
- Local cloud sectioning
- Core system usage,
- Application system design,

SoLCDD, LCD, SysD, SD

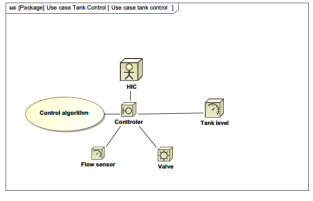
Procurement of:
- Application hardware, OS, Router,
- Installing OS

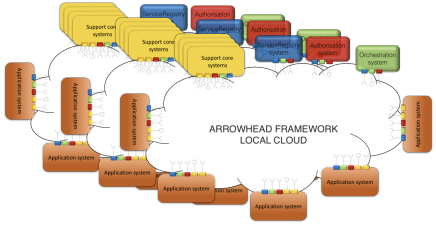
White box engineering of:
- Application systems and services code
- Orchestration and security policys
- Installing core and application systems to procured HW
- Configuration of network,

LCDD, SysDD, IDD

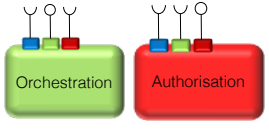
Deployment of:
- HW with core and application systems in plant,
- Orchestration policys
- Security policys

Commissioning of:
- Local cloud functionality
- System of local cloud functionality

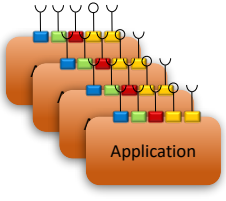




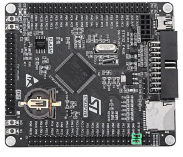
Core system selection



Application system & service design

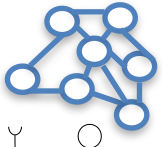


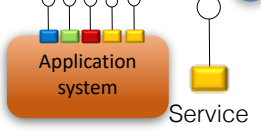
Devices and network



OS

Network

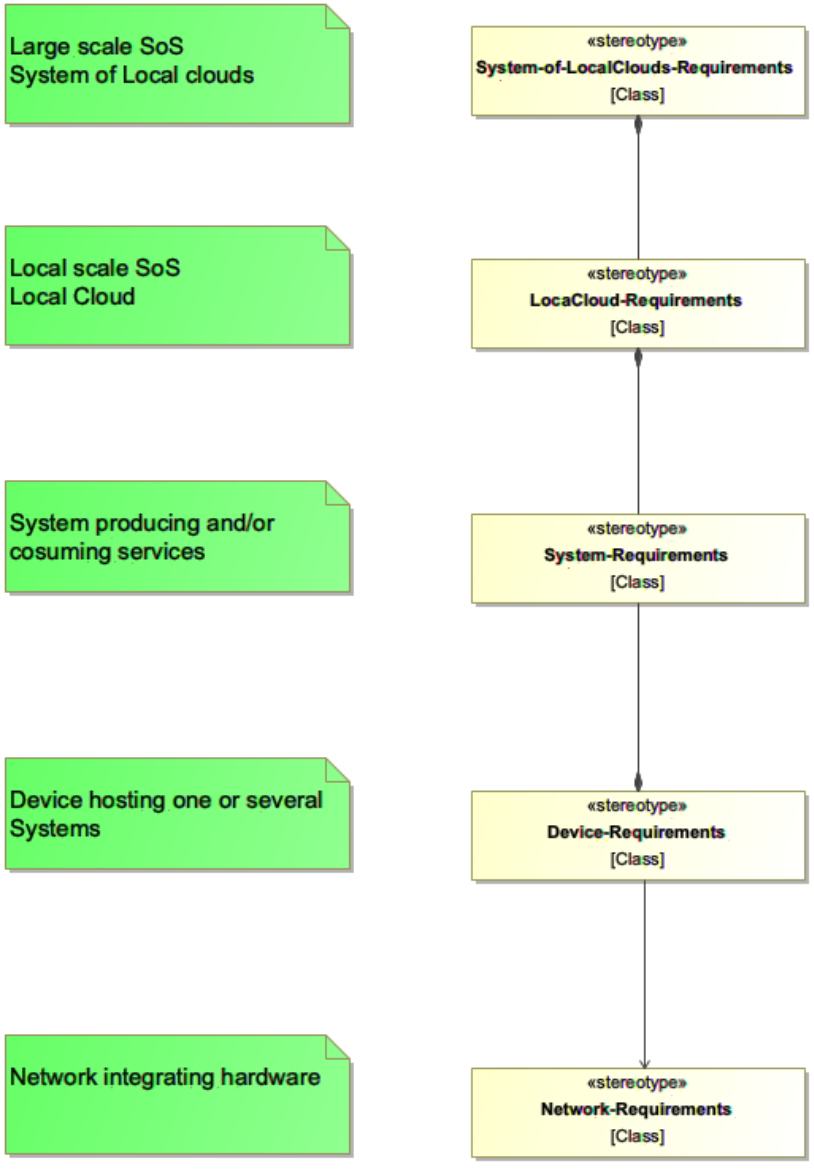




Physical deployment at site
- **Devices**
- **Routers**
- **Power supply**
- **Network connection**
-

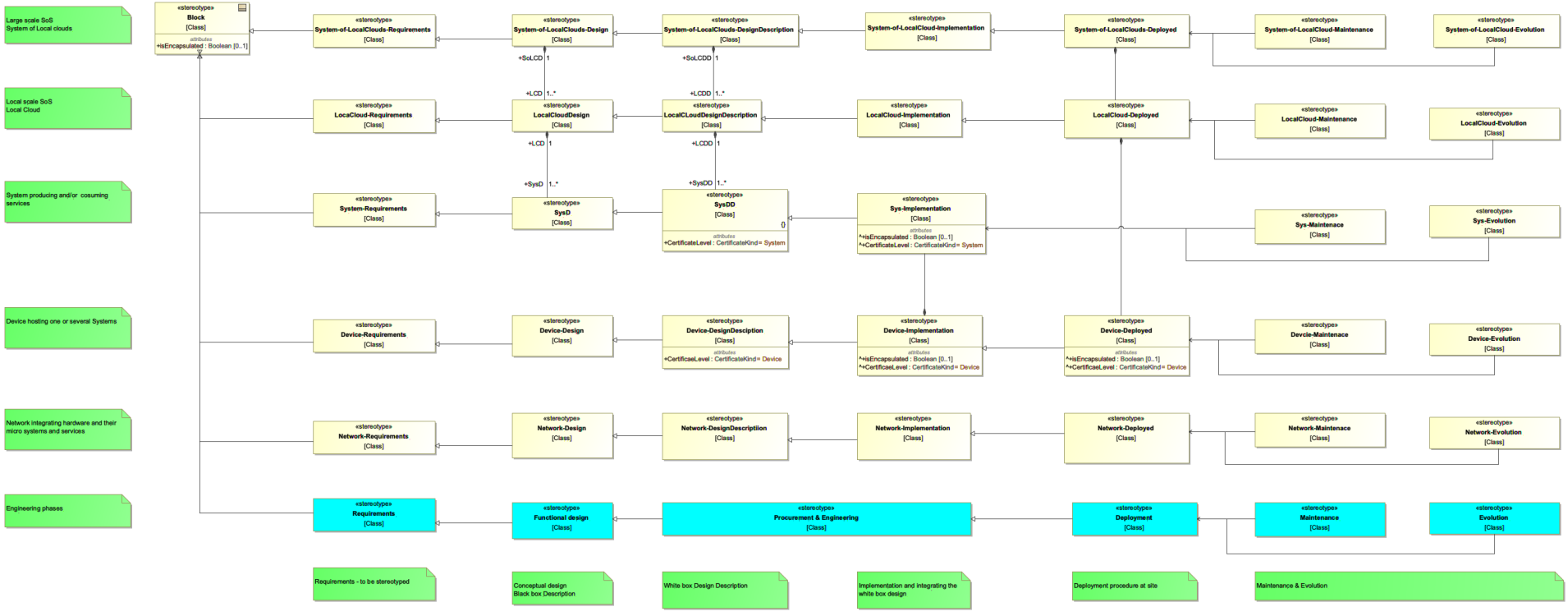
ARROWHEAD TOOLS

Architecture



ARROWHEAD TOOLS

SoS architecture and engineering in SysML

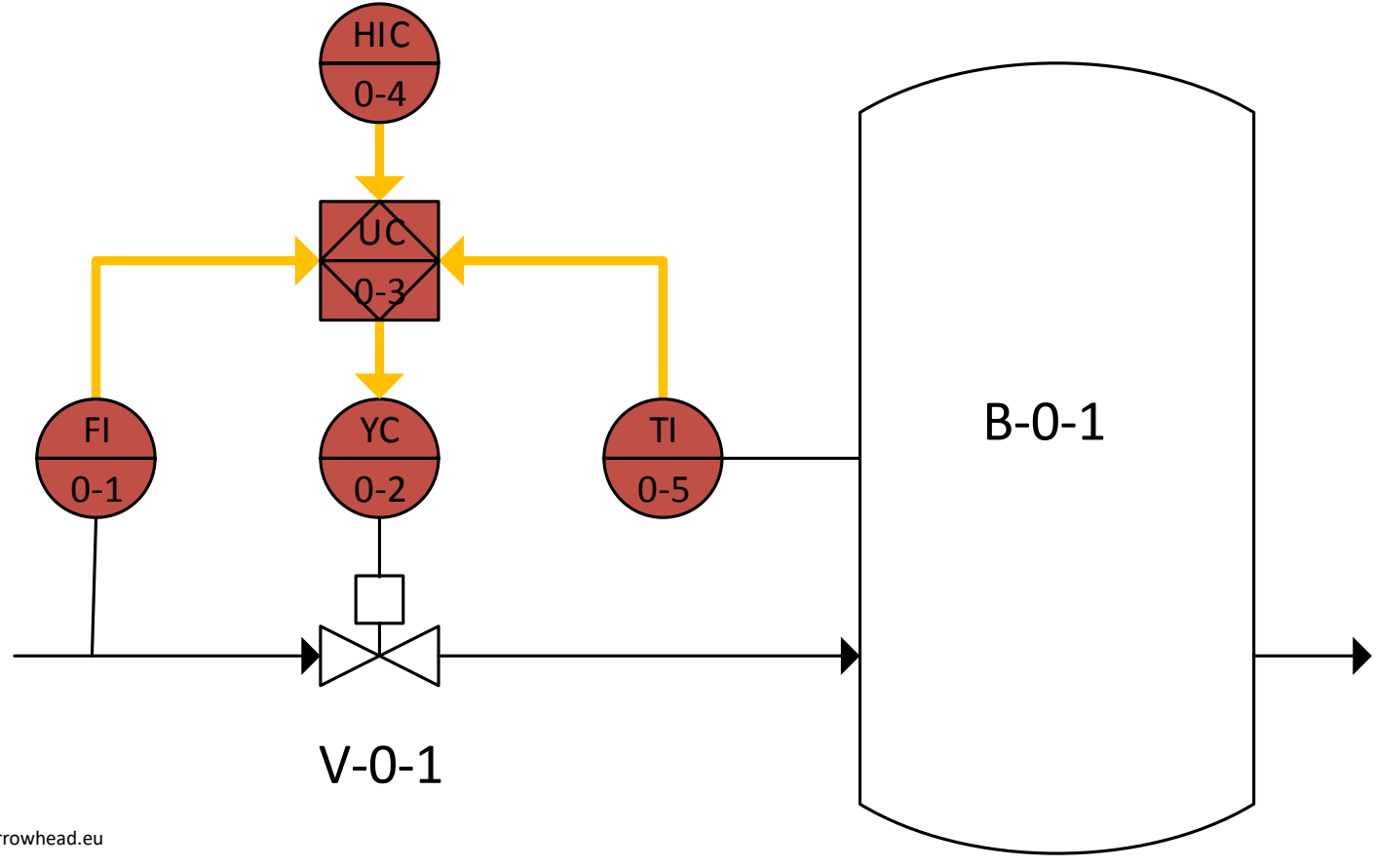


ARROWHEAD TOOLS

Lets start with a use case

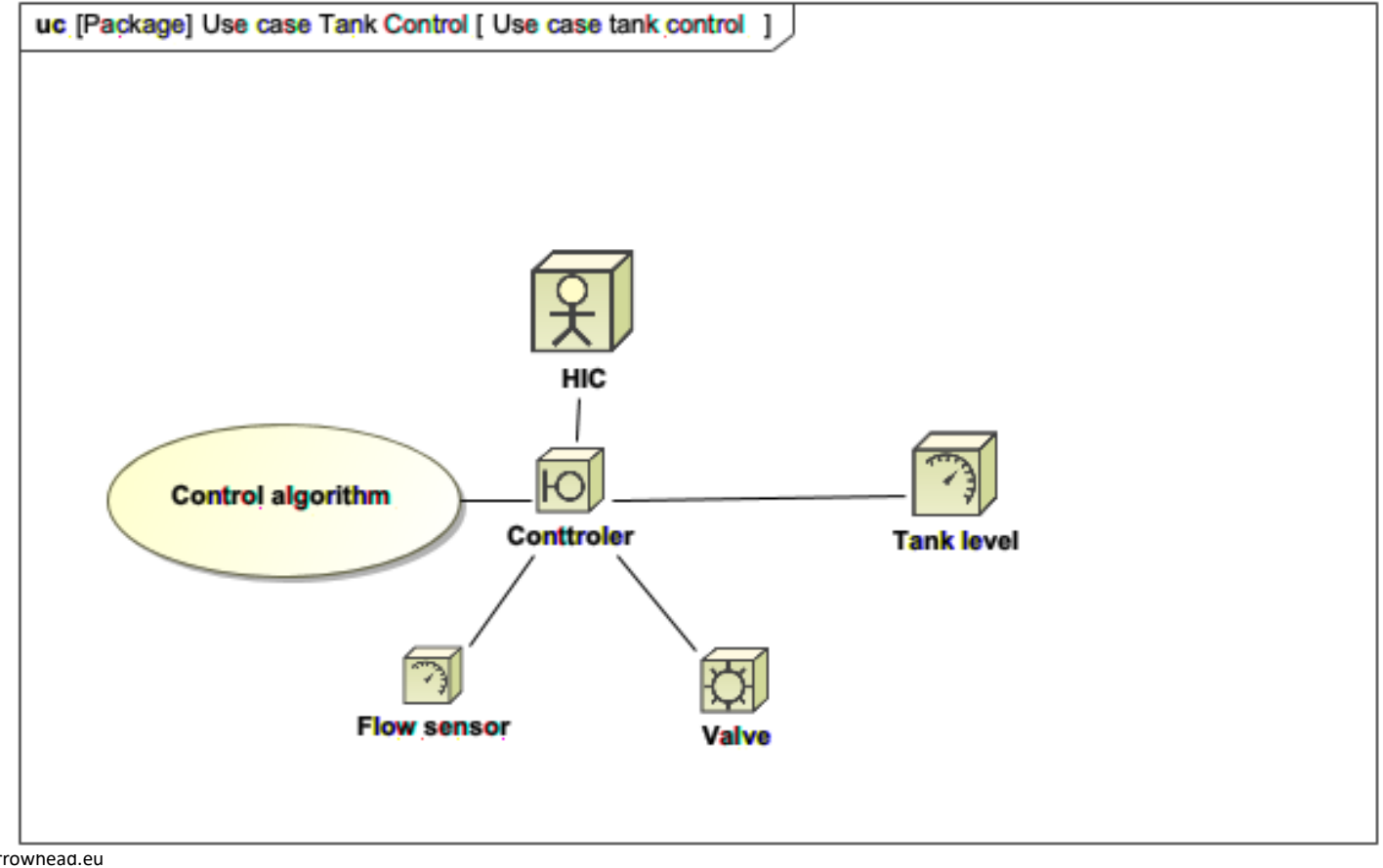
ARROWHEAD TOOLS

Lets start with a use case



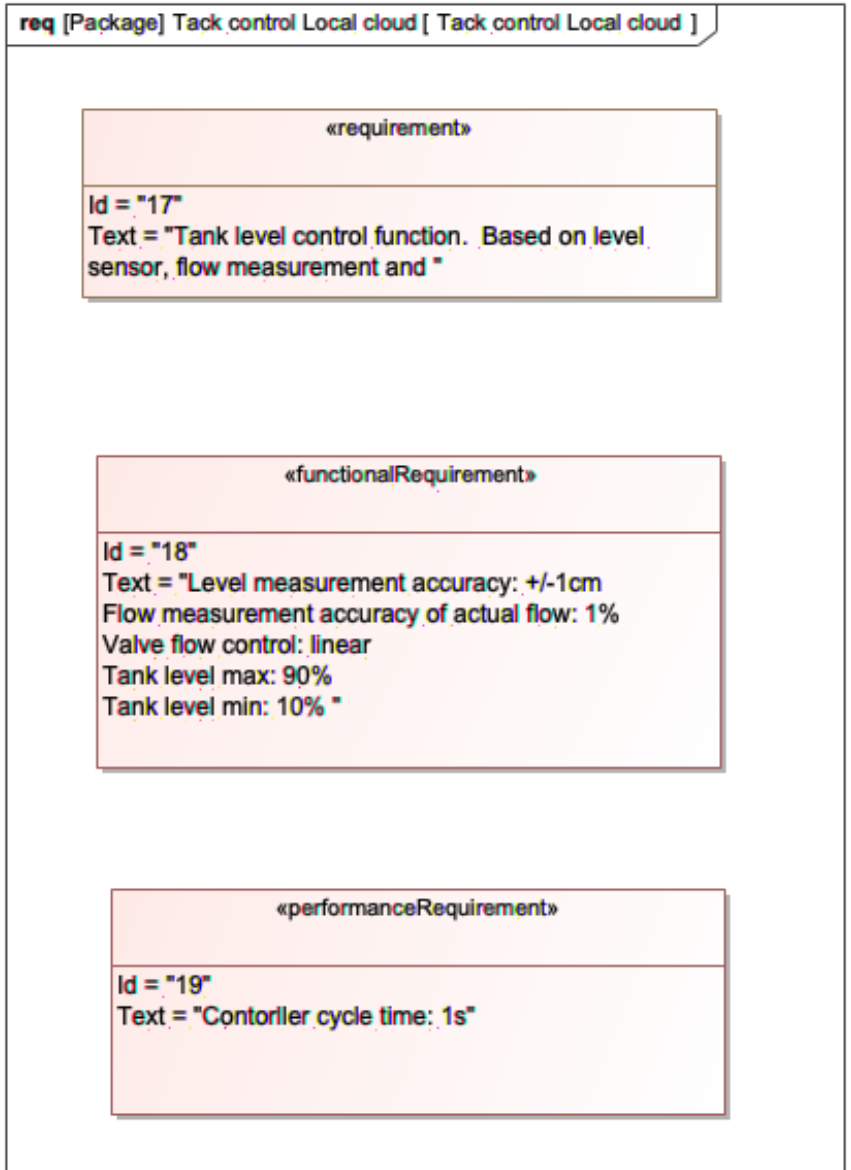
ARROWHEAD TOOLS

Use case diagram

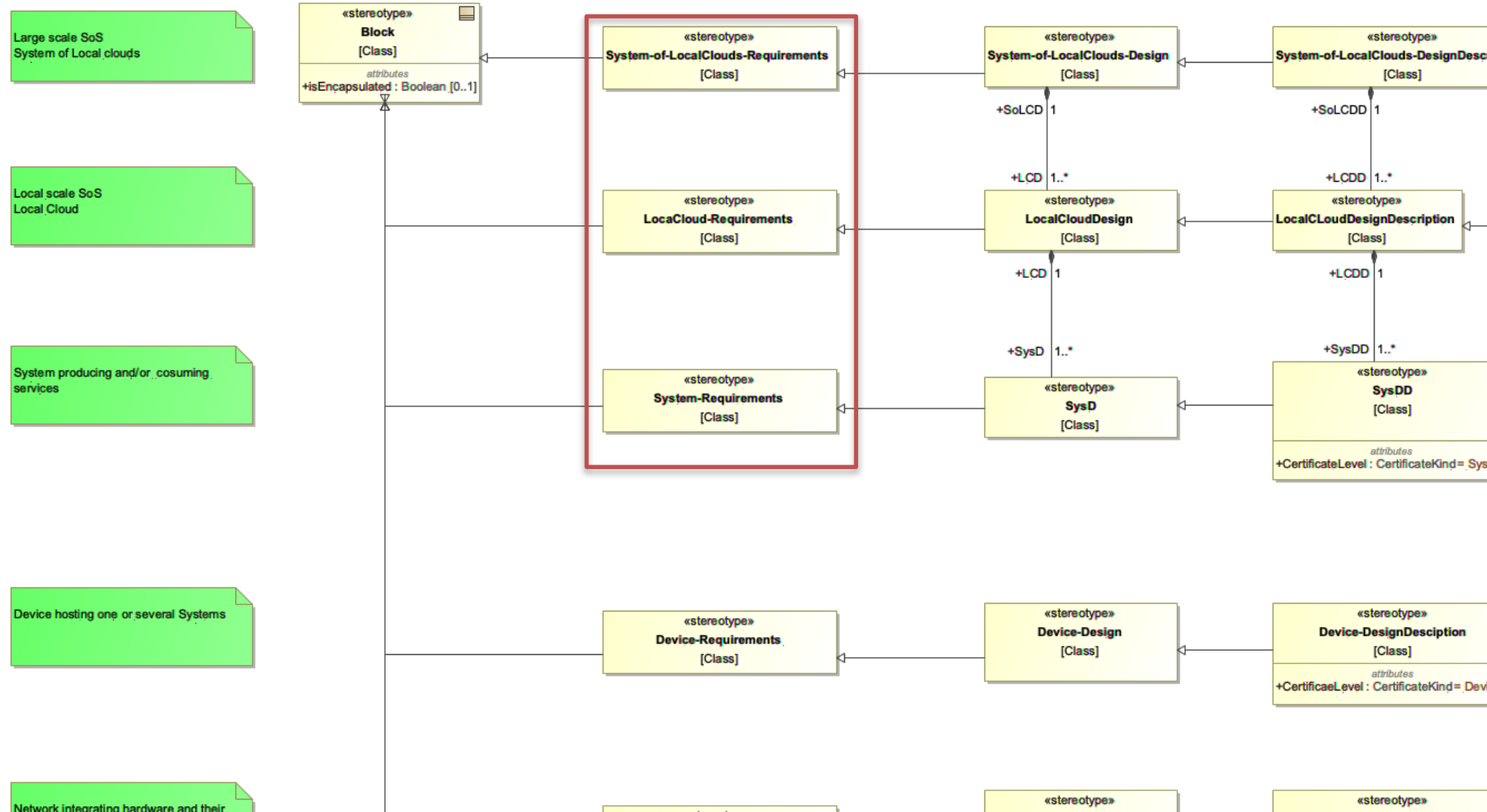


ARROWHEAD TOOLS

Requirements



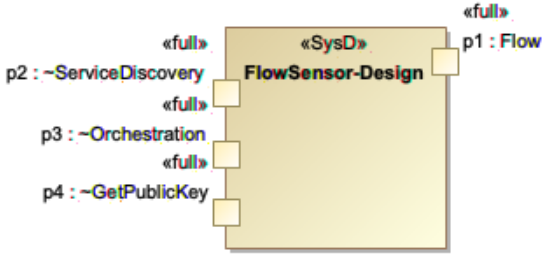
SoS architecture and engineering in SysML

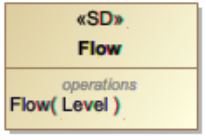


Functional system and service design

Micro-systems

Micro-services

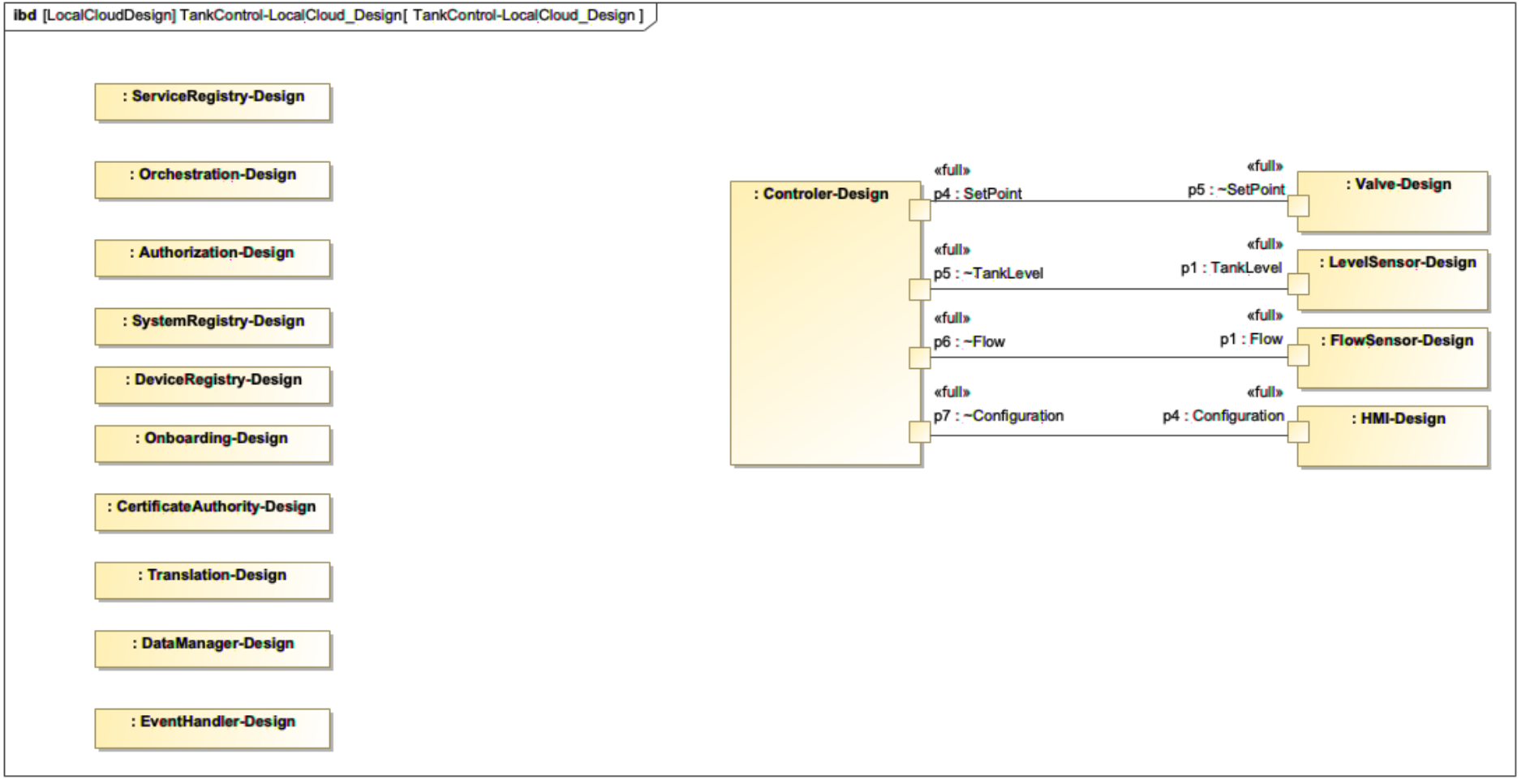




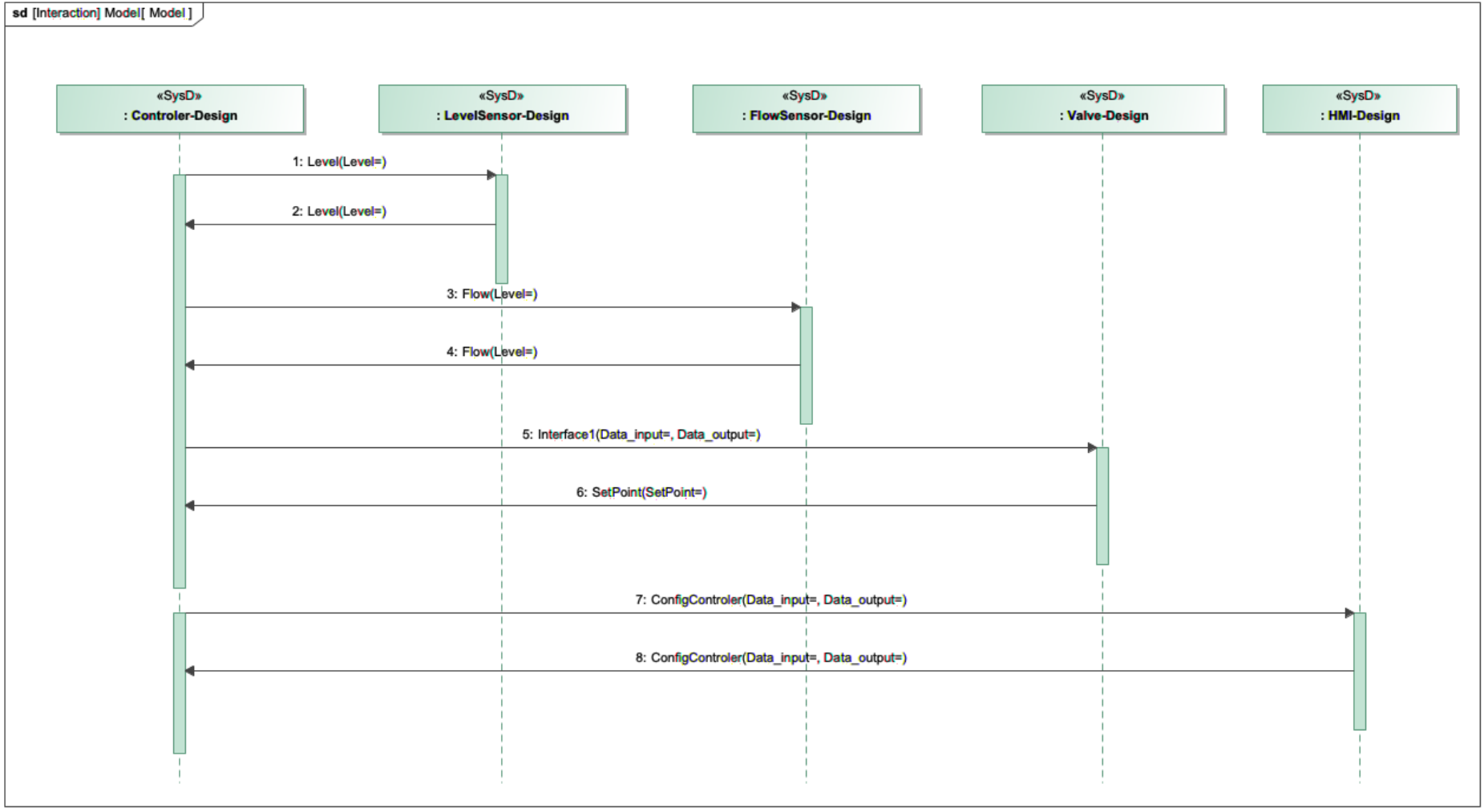
ARROWHEAD TOOLS

Functional SoS design - black box

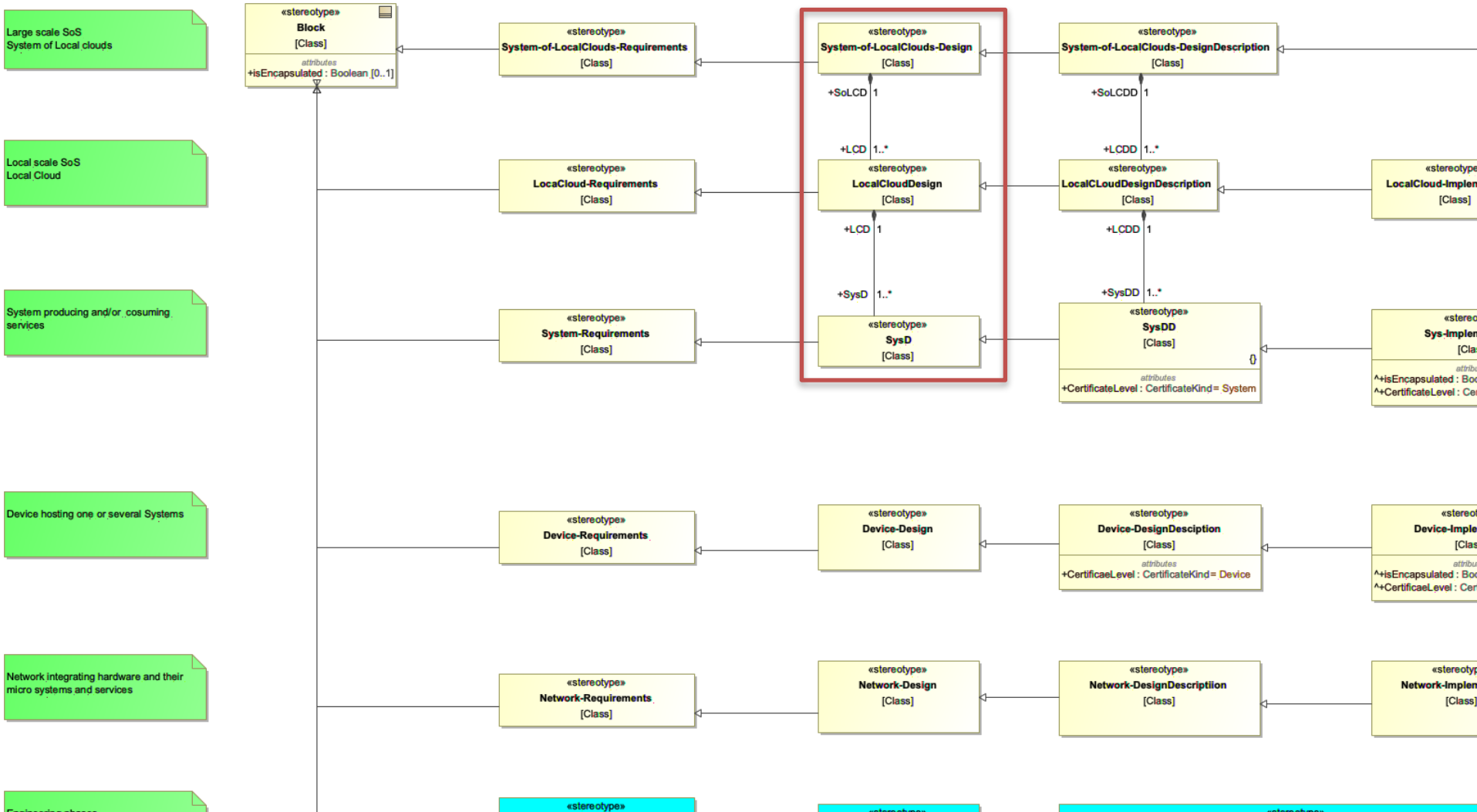
Local cloud functional orchestration



Service exchange functionality



SoS architecture and engineering in SysML



White box engineering

SysDD and IDD

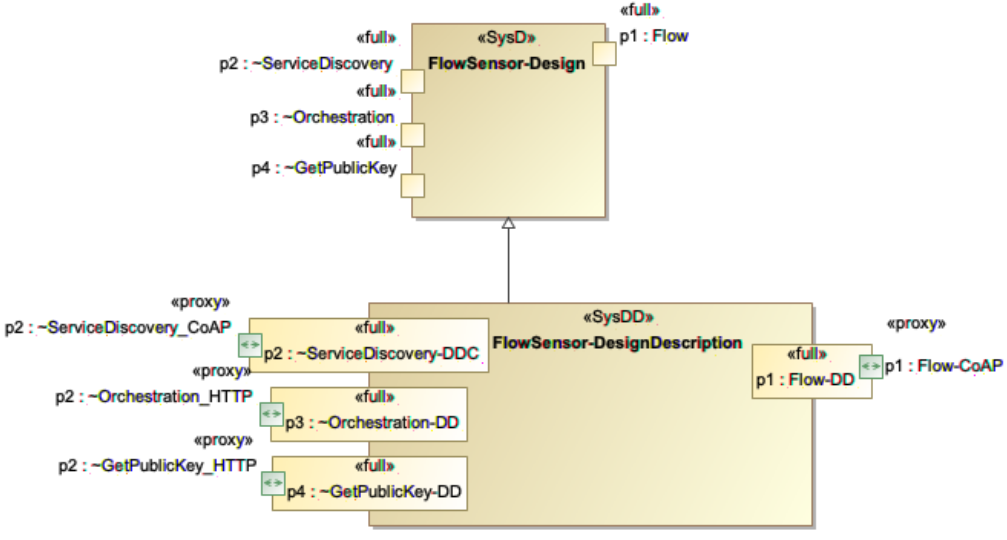
www.arrowhead.eu

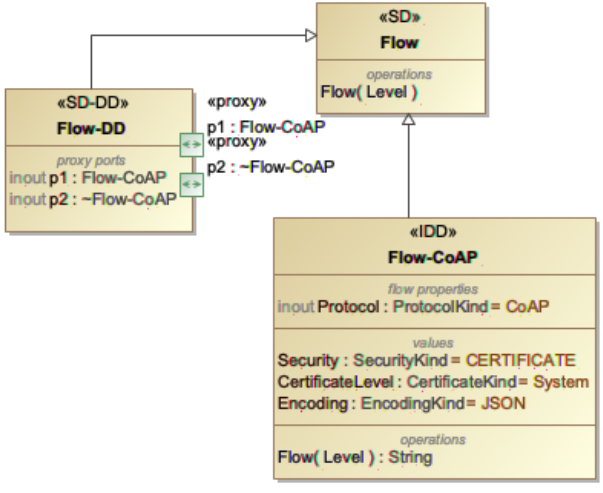
ARROWHEAD TOOLS

Functional system and service design & design description/implementation black box & white box + code

Micro-systems

Micro-services

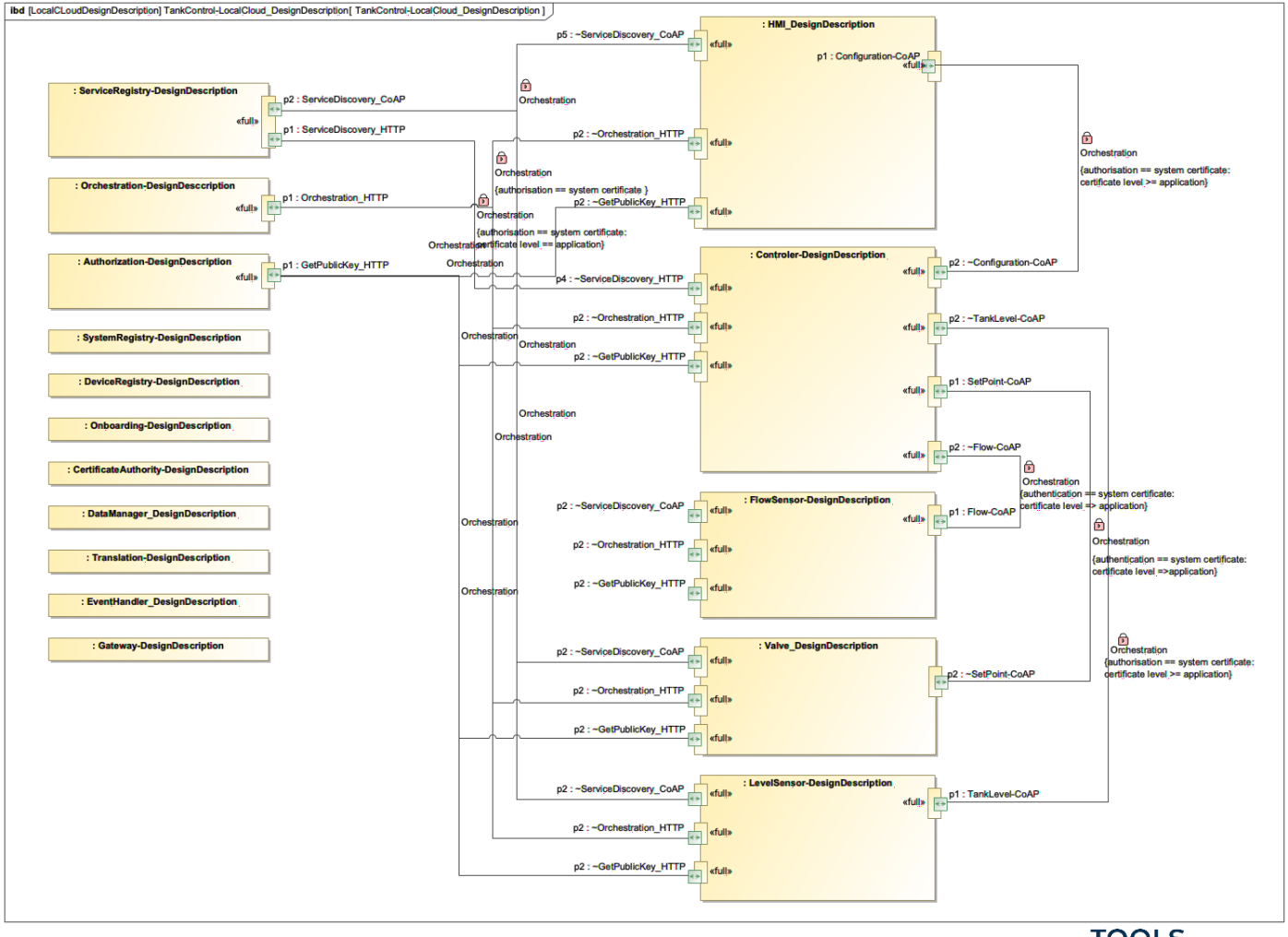




ARROWHEAD TOOLS

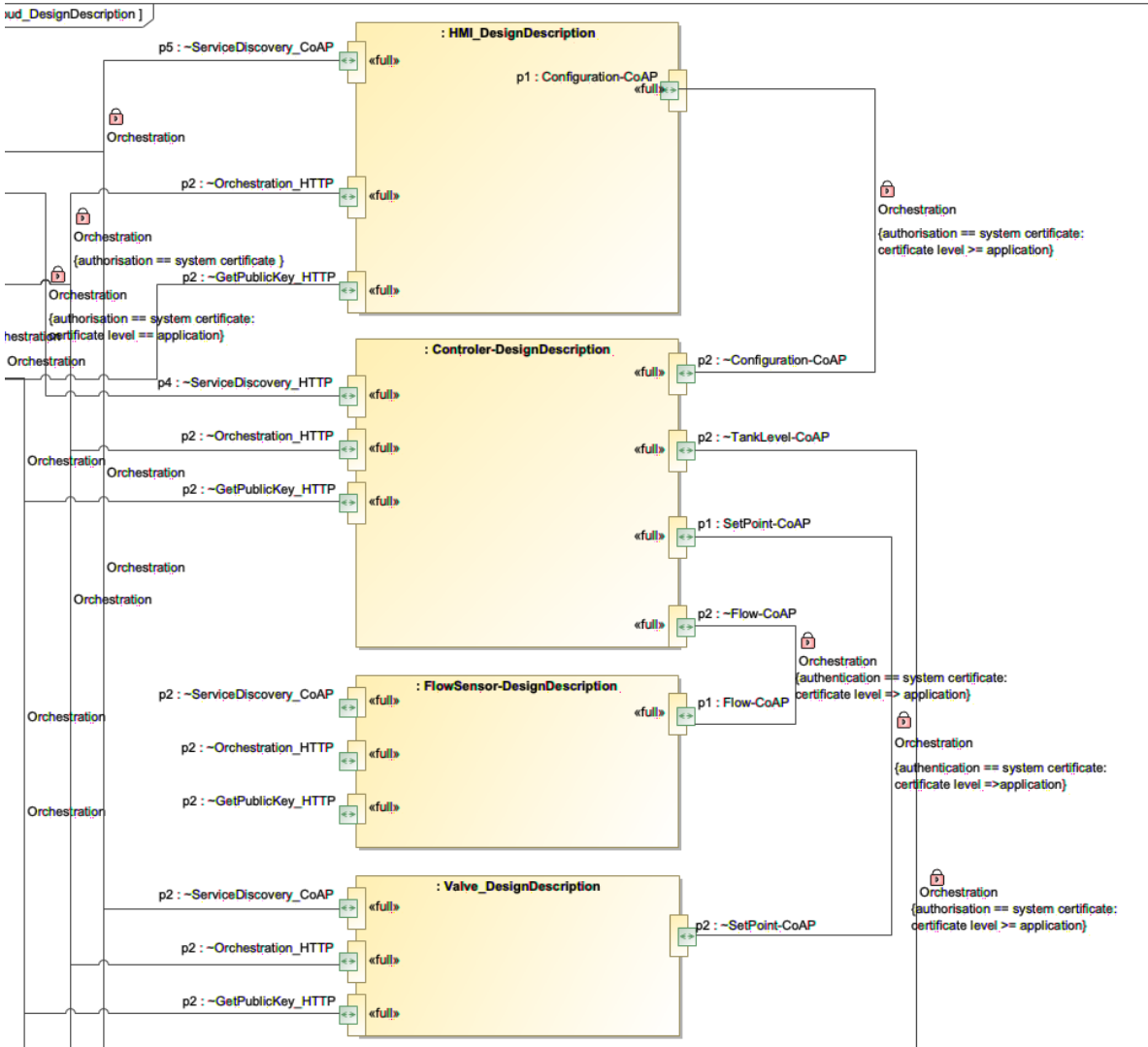
Functional LC design description - white box

Local cloud functional design description model



Functional LC design description - white box

Local cloud functional design description model with **security policys**



Orchestration policys - rules and conditions

#	Name	Role (Connector End A)	Role (Connector End B)
1	Orchestration	inout p4 : ~ServiceDiscovery_HTTP	inout p1 : ServiceDiscovery_HTTP
2	Orchestration	inout p2 : ~Orchestration_HTTP	inout p1 : Orchestration_HTTP
3	Orchestration	inout p2 : ~GetPublicKey_HTTP	inout p1 : GetPublicKey_HTTP
4	Orchestration	inout p1 : Flow-CoAP	inout p2 : ~Flow-CoAP
5	Orchestration	inout p5 : ~ServiceDiscovery_CoAP	inout p2 : ServiceDiscovery_CoAP
6	Orchestration	inout p2 : ~Orchestration_HTTP	inout p1 : Orchestration_HTTP
7	Orchestration	inout p2 : ~GetPublicKey_HTTP	inout p1 : GetPublicKey_HTTP
8	Orchestration	inout p2 : ~Configuration-CoAP	inout p2 : ~Configuration-CoAP
9	Orchestration	inout p2 : ~ServiceDiscovery_CoAP	inout p2 : ServiceDiscovery_CoAP
10	Orchestration	inout p2 : ~GetPublicKey_HTTP	inout p1 : GetPublicKey_HTTP
11	Orchestration	inout p2 : ~ServiceDiscovery_CoAP	inout p2 : ServiceDiscovery_CoAP
12	Orchestration	inout p2 : ~Orchestration_HTTP	inout p1 : Orchestration_HTTP
13	Orchestration	inout p2 : ~GetPublicKey_HTTP	inout p1 : GetPublicKey_HTTP
14	Orchestration	inout p2 : ~SetPoint-CoAP	inout p1 : SetPoint-CoAP

ARROWHEAD TOOLS

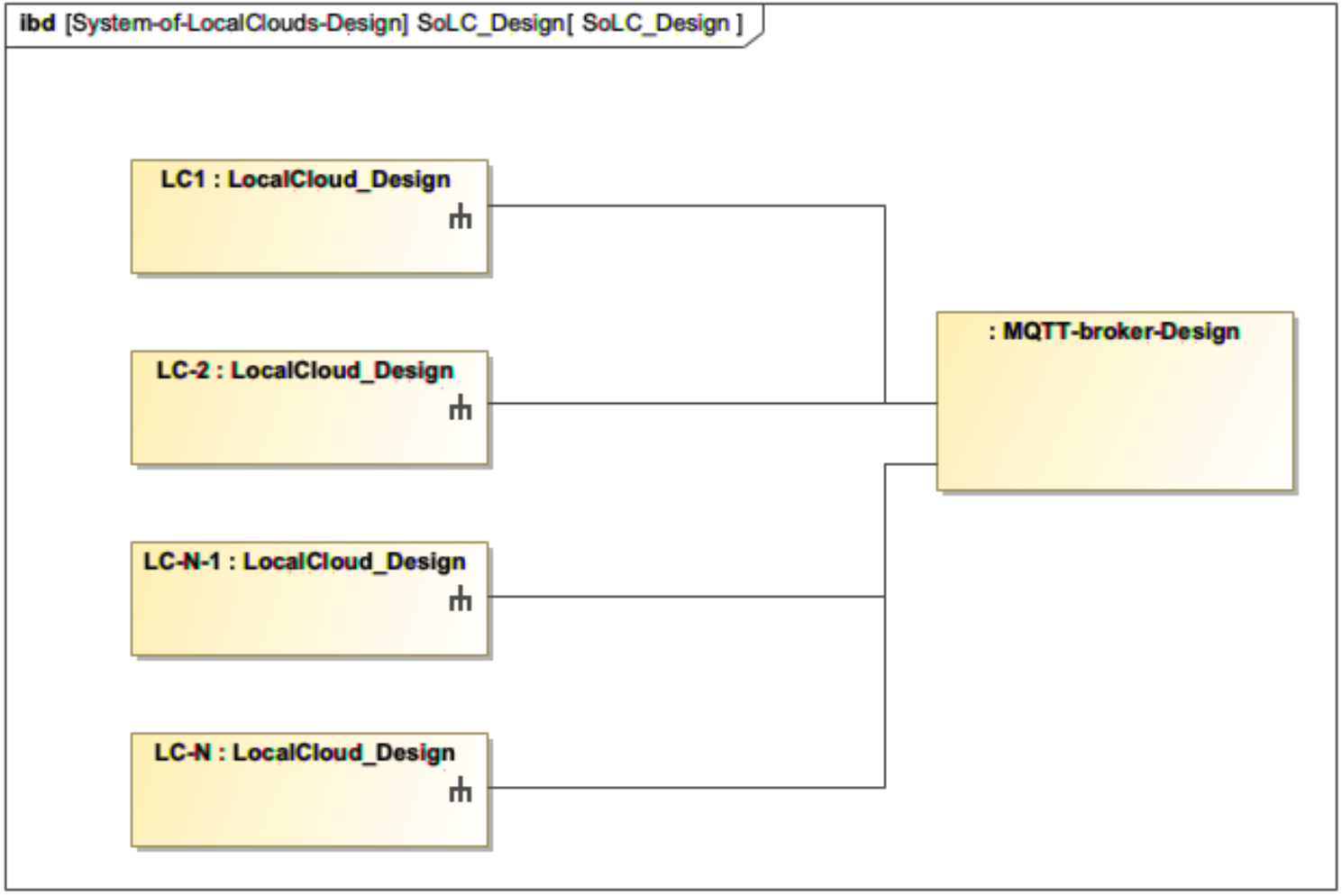
Security policys - rules and conditions

#	Name	Role	Role	Security constrains
1	Orchestration	inout p4 : ~ServiceDiscovery_HTTP	inout p1 : ServiceDiscovery_HTTP	{} Security p=authorisation == system certificate: ce...
2	Orchestration	inout p5 : ~ServiceDiscovery_CoAP	inout p2 : ServiceDiscovery_CoAP	
3	Orchestration	inout p2 : ~Orchestration_HTTP	inout p1 : Orchestration_HTTP	{} Security-3=authorisation == system certificate
4	Orchestration	inout p1 : Flow-CoAP	inout p2 : ~Flow-CoAP	{} Secuirty policy=authentication == system certifica...

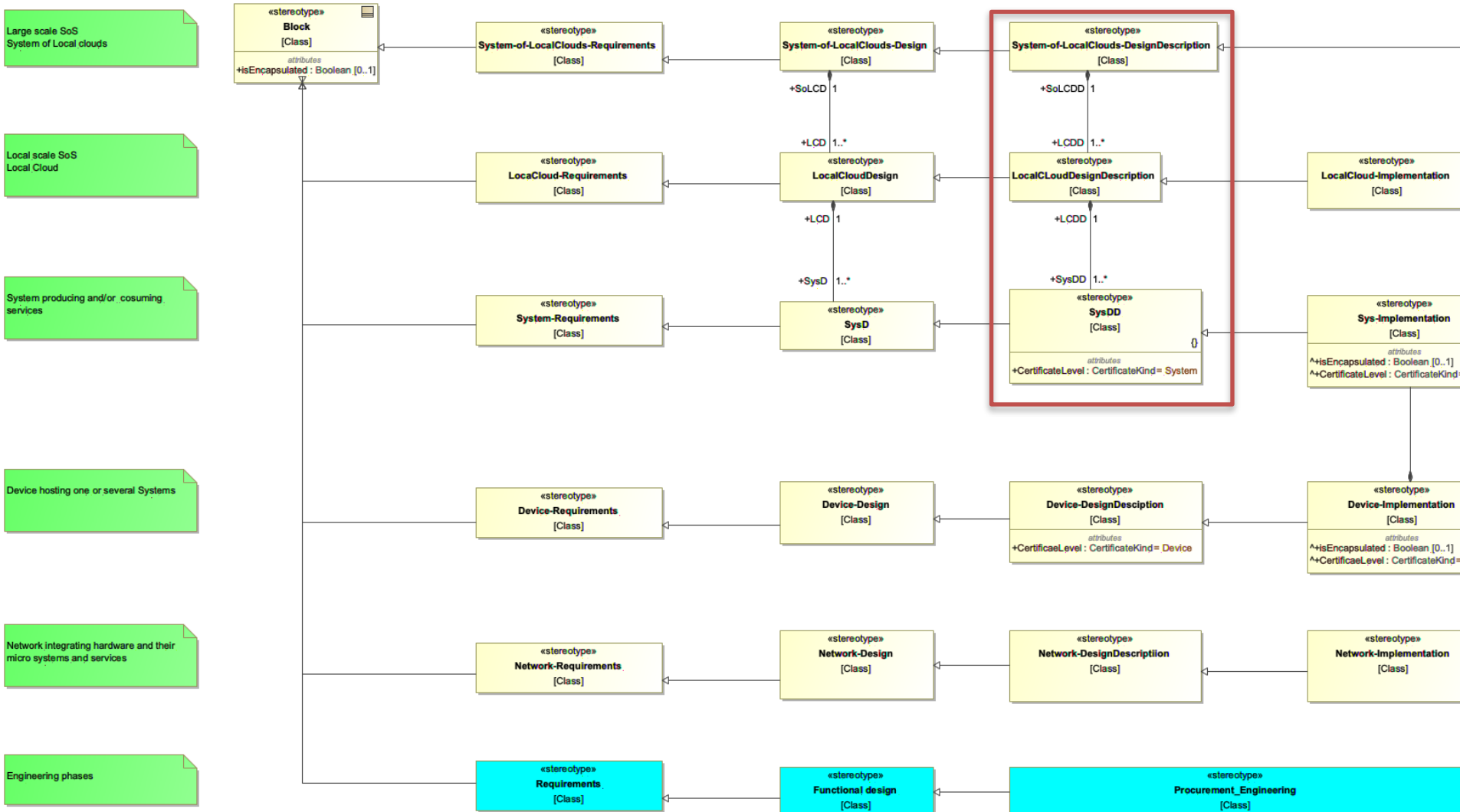
ARROWHEAD TOOLS

Functional SoLC design

System o local clouds functional design model



SoS architecture and engineering in SysML



Implementation

We also need

- Devices
- Network

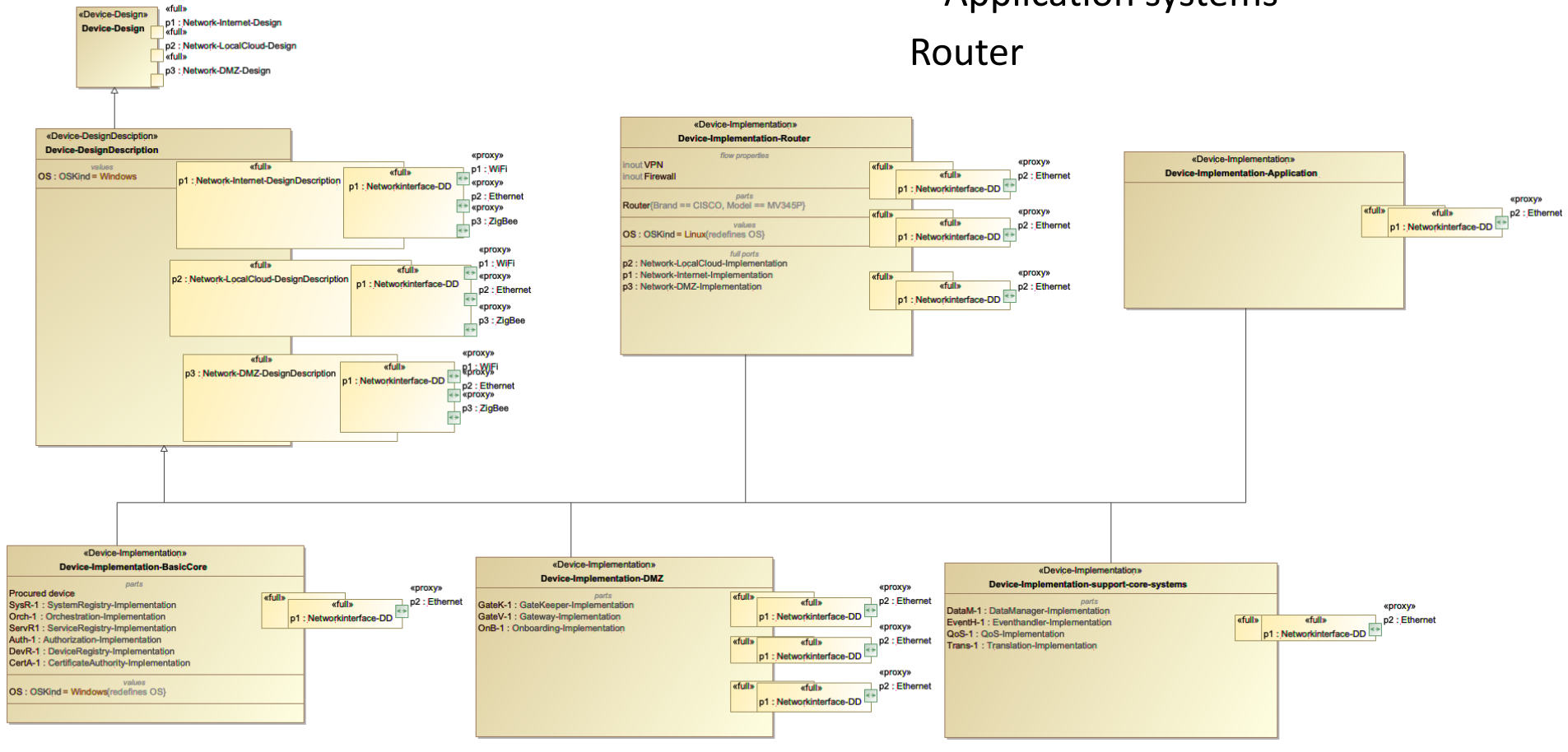
ARROWHEAD TOOLS

Device implementation

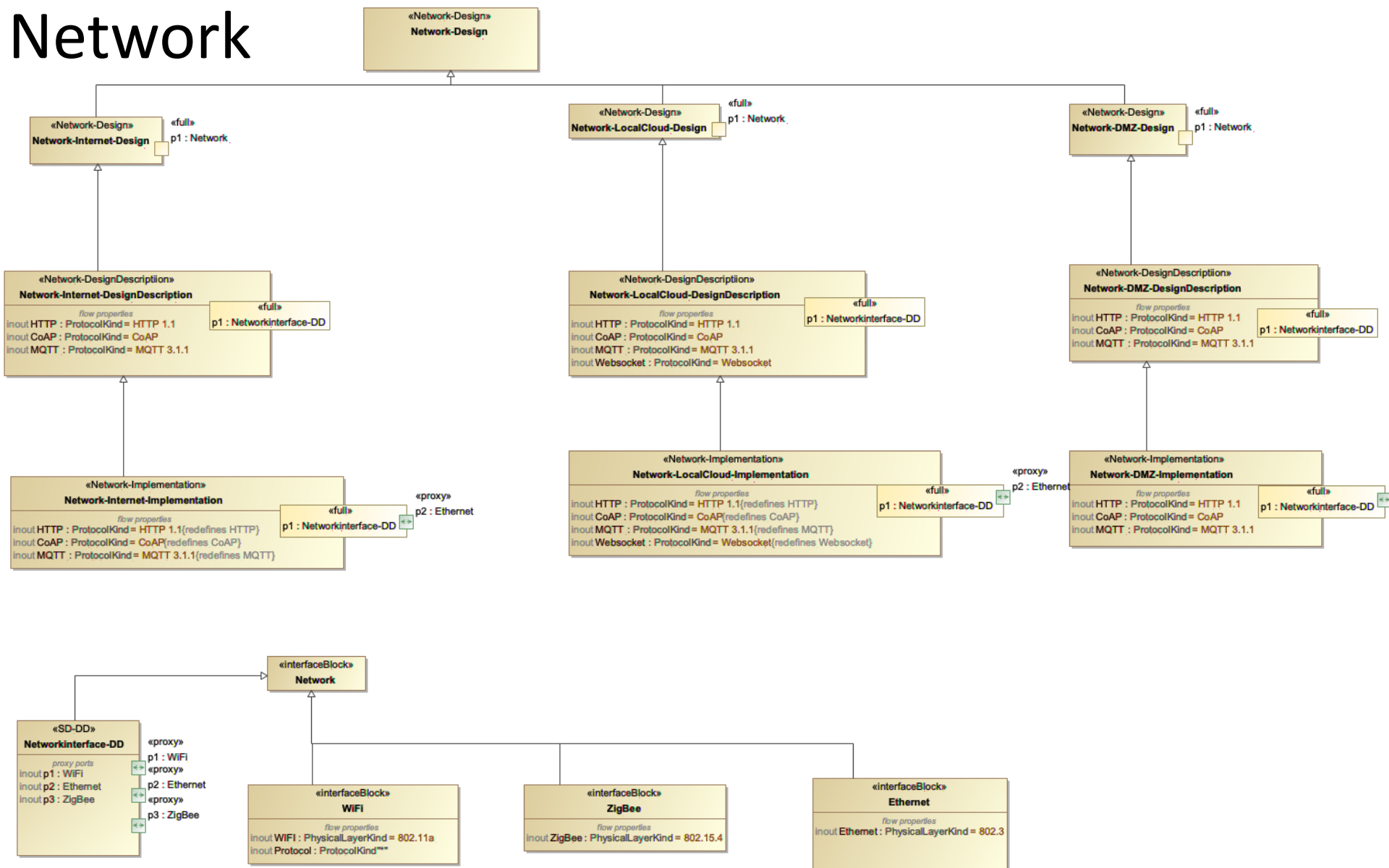
Devices with

- Mandatory core systems
- Support core systems
- Application systems

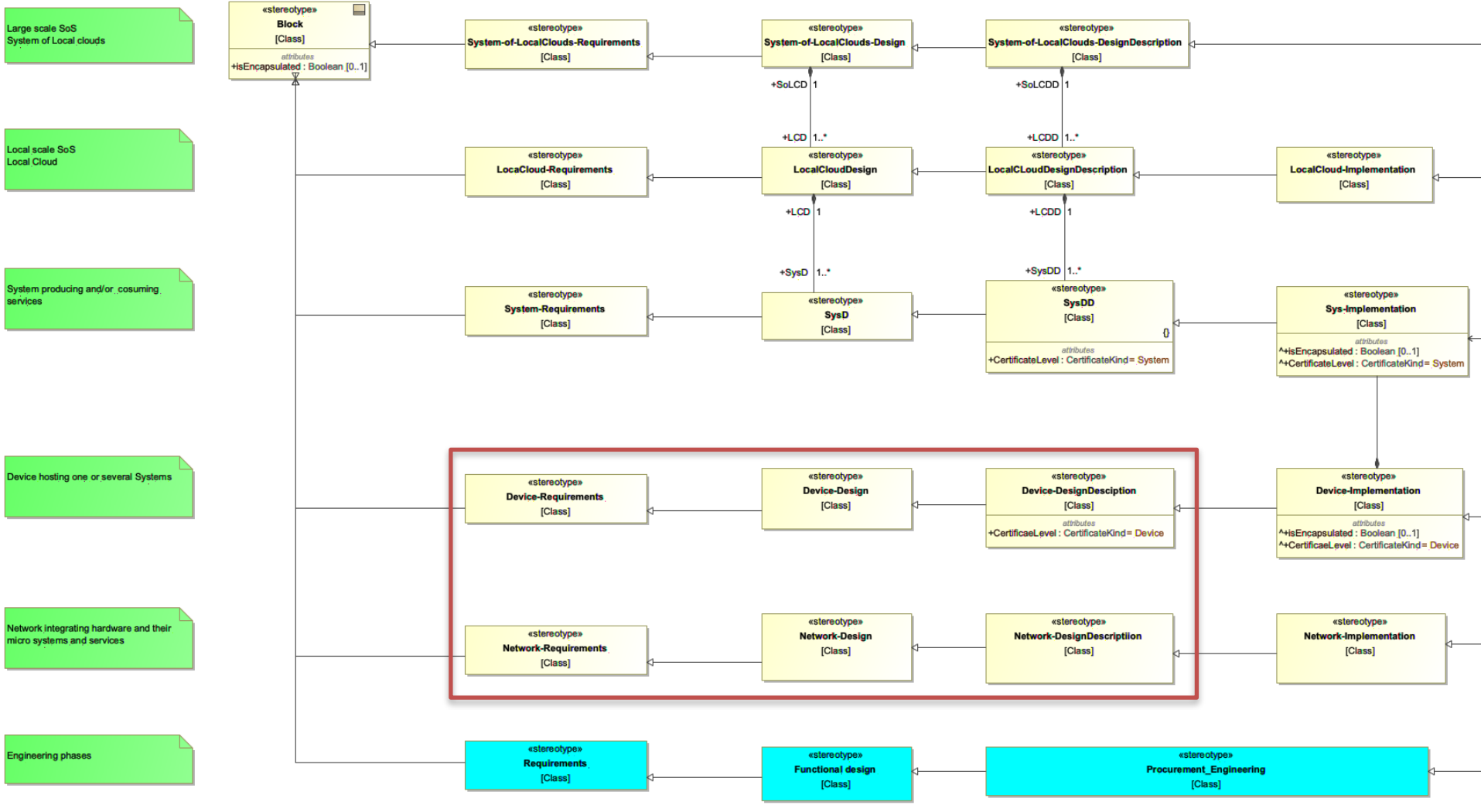
Router



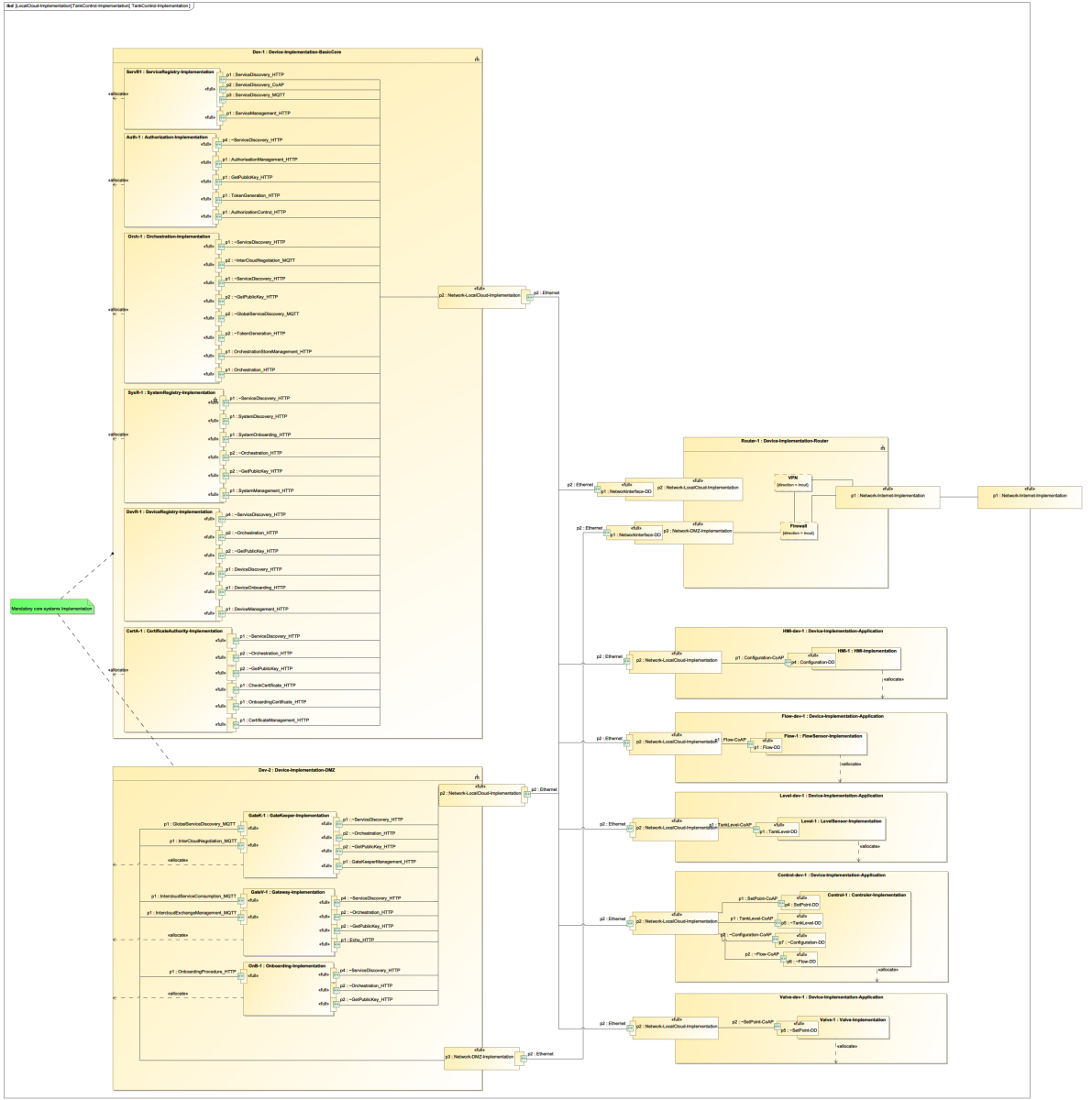
Network


«Network-Design»
Network-Design
«Network-Design»
Network-Internet-Design
«full»
p1 : Network
«Network-Design»
Network-LocalCloud-Design
«full»
p1 : Network
«Network-Design»
Network-DMZ-Design
«full»
p1 : Network
«Network-DesignDescriptiion»
Network-Internet-DesignDescription
flow properties
inout HTTP : ProtocolKind = HTTP 1.1
inout CoAP : ProtocolKind = CoAP
inout MQTT : ProtocolKind = MQTT 3.1.1
«full»
p1 : Networkinterface-DD
«Network-DesignDescriptiion»
Network-LocalCloud-DesignDescription
flow properties
inout HTTP : ProtocolKind = HTTP 1.1
inout CoAP : ProtocolKind = CoAP
inout MQTT : ProtocolKind = MQTT 3.1.1
inout Websocket : ProtocolKind = Websocket
«full»
p1 : Networkinterface-DD
«Network-DesignDescriptiion»
Network-DMZ-DesignDescription
flow properties
inout HTTP : ProtocolKind = HTTP 1.1
inout CoAP : ProtocolKind = CoAP
inout MQTT : ProtocolKind = MQTT 3.1.1
«full»
p1 : Networkinterface-DD
«Network-Implementation»
Network-Internet-Implementation
flow properties
inout HTTP : ProtocolKind = HTTP 1.1{redefines HTTP}
inout CoAP : ProtocolKind = CoAP{redefines CoAP}
inout MQTT : ProtocolKind = MQTT 3.1.1{redefines MQTT}
«full»
p1 : Networkinterface-DD
«proxy»
p2 : Ethernet
«Network-Implementation»
Network-LocalCloud-Implementation
flow properties
inout HTTP : ProtocolKind = HTTP 1.1{redefines HTTP}
inout CoAP : ProtocolKind = CoAP{redefines CoAP}
inout MQTT : ProtocolKind = MQTT 3.1.1{redefines MQTT}
inout Websocket : ProtocolKind = Websocket{redefines Websocket}
«full»
p1 : Networkinterface-DD
«proxy»
p2 : Ethernet
«Network-Implementation»
Network-DMZ-Implementation
flow properties
inout HTTP : ProtocolKind = HTTP 1.1
inout CoAP : ProtocolKind = CoAP
inout MQTT : ProtocolKind = MQTT 3.1.1
«full»
p1 : Networkinterface-DD
«interfaceBlock»
Network
«SD-DD»
Networkinterface-DD
proxy ports
inout p1 : WiFi
inout p2 : Ethernet
inout p3 : ZigBee
«proxy»
p1 : WiFi
«proxy»
p2 : Ethernet
«proxy»
p3 : ZigBee
«interfaceBlock»
WiFi
flow properties
inout WIFI : PhysicalLayerKind = 802.11a
inout Protocol : ProtocolKind""
«interfaceBlock»
ZigBee
flow properties
inout ZigBee : PhysicalLayerKind = 802.15.4
«interfaceBlock»
Ethernet
flow properties
inout Ethernet : PhysicalLayerKind = 802.3

SoS architecture and engineering in SysML



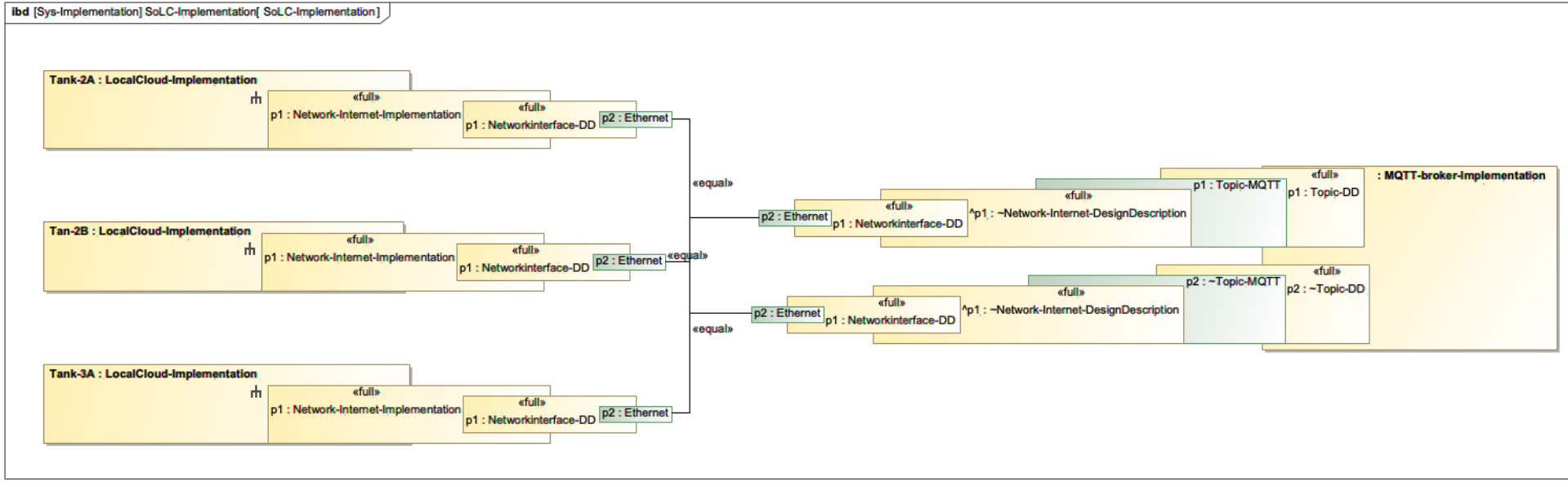
Functional SoS/ Local cloud implementation engineering



www.arrowhead.eu

Functional SoLC implementation engineering

System o local clouds functional implementation model



ARROWHEAD TOOLS

Extraction of code

#	Name	Specification	Constrained Element
1	{} Implementation	Authorization v4.3.0 == https://github.com/eclipse-arrowhead/core-java-spring/tree/master/authorization	Authorization-Implementation
2	{} Implementation	CertificateAuthority v4.3.0 == https://github.com/eclipse-arrowhead/core-java-spring/tree/master/certificate-authority	CertificateAuthority-Implementation
3	{} Implementation	v4.3.0 == http://github.com/eclipse-arrowhead/core-java-spring/datamanager	DataManager-Implementation
4	{} Implementation	DeviceRegistry v4.3.0 == https://github.com/eclipse-arrowhead/core-java-spring/tree/master/deviceregistry	DeviceRegistry-Implementation
5	{} Implementation	v4.3.0 == https://github.com/eclipse-arrowhead/core-java-spring/tree/master/eventhandler	Eventhandler-Implementation
6	{} Implementation	GateKeeper v4.3.0 == https://github.com/eclipse-arrowhead/core-java-spring/tree/master/gatekeeper	GateKeeper-Implementation
7	{} Implementation	Gateway v4.3.0 == https://github.com/eclipse-arrowhead/core-java-spring/tree/master/gateway	Gateway-Implementation
8	{} Implementation	v4.3.0 ==http://www.github.com/eclipse-arrowhead/mqtt-broker	MQTT-broker-Implementation
9	{} Implementation	Onboarding v4.3.0 == https://github.com/eclipse-arrowhead/core-java-spring/tree/master/onboarding	Onboarding-Implementation
10	{} Implementation	Otchestration v4.3.0 == https://github.com/eclipse-arrowhead/core-java-spring/tree/master/orchestrator	Orchestration-Implementation
11	{} Implementation	v4.3.0 == https://github.com/eclipse-arrowhead/core-java-spring/tree/master/qos-monitor	QoS-Implementation
12	{} Implementation	ServiceRegistry v4.3.0 == https://github.com/eclipse-arrowhead/core-java-spring/tree/master/serviceregistry	ServiceRegistry-Implementation
13	{} Implementing code pack	SystemRegistry v4.3.0 == https://github.com/eclipse-arrowhead/core-java-spring/tree/master/systemregistry	SystemRegistry-Implementation
14	{} Implementation	v4.3.0 == https://www.github.com/eclipse-arrowhead/core-java-spring/translation	Translation-Implementation

Move from here to Docker containers for deployment to

Selected HW and OS

Server - Linux - Ubuntu 20.10

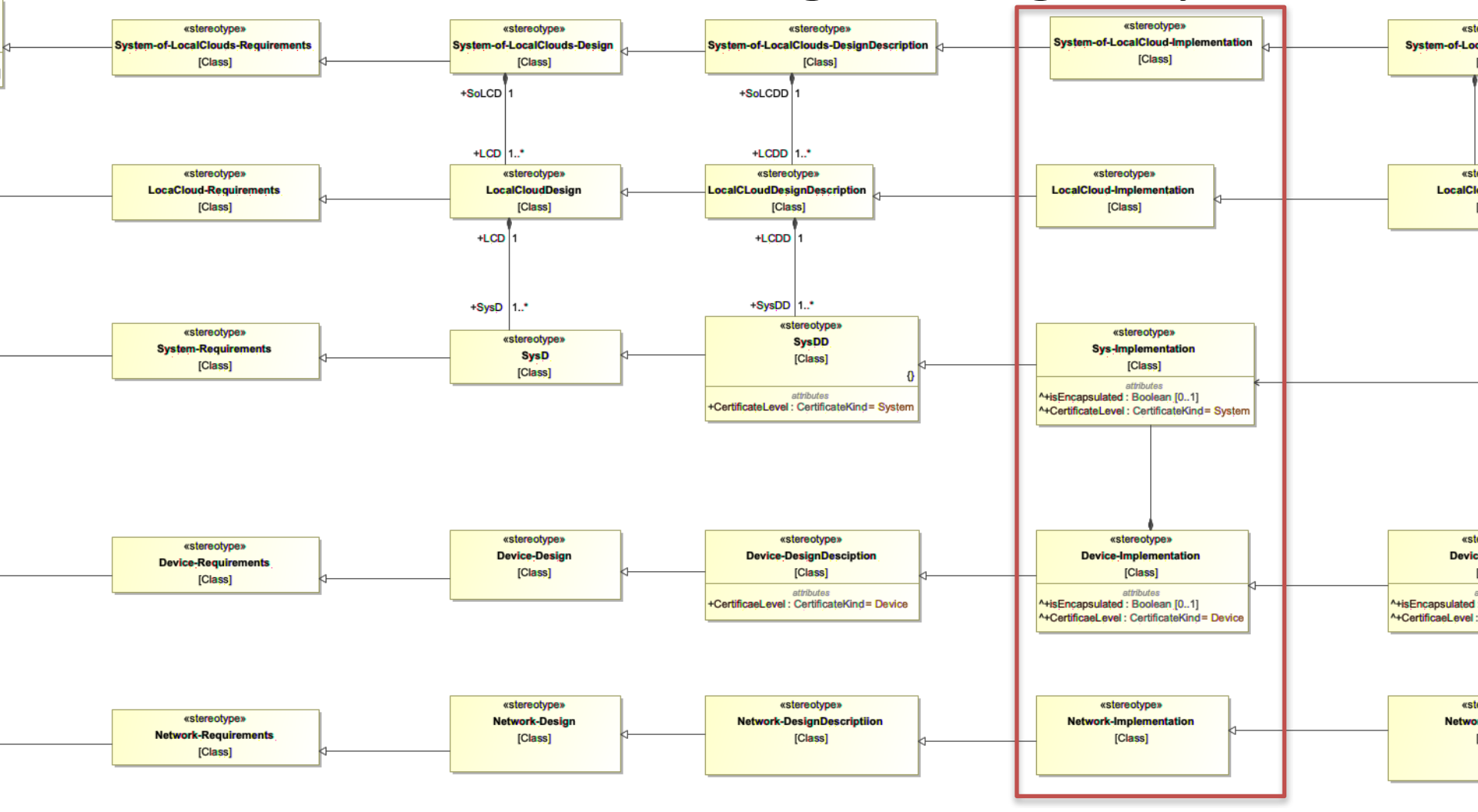
Desktop computer, Windows 10.xx, OSX 11.2.1

Embedded system

Raspberry PI

ARROWHEAD TOOLS

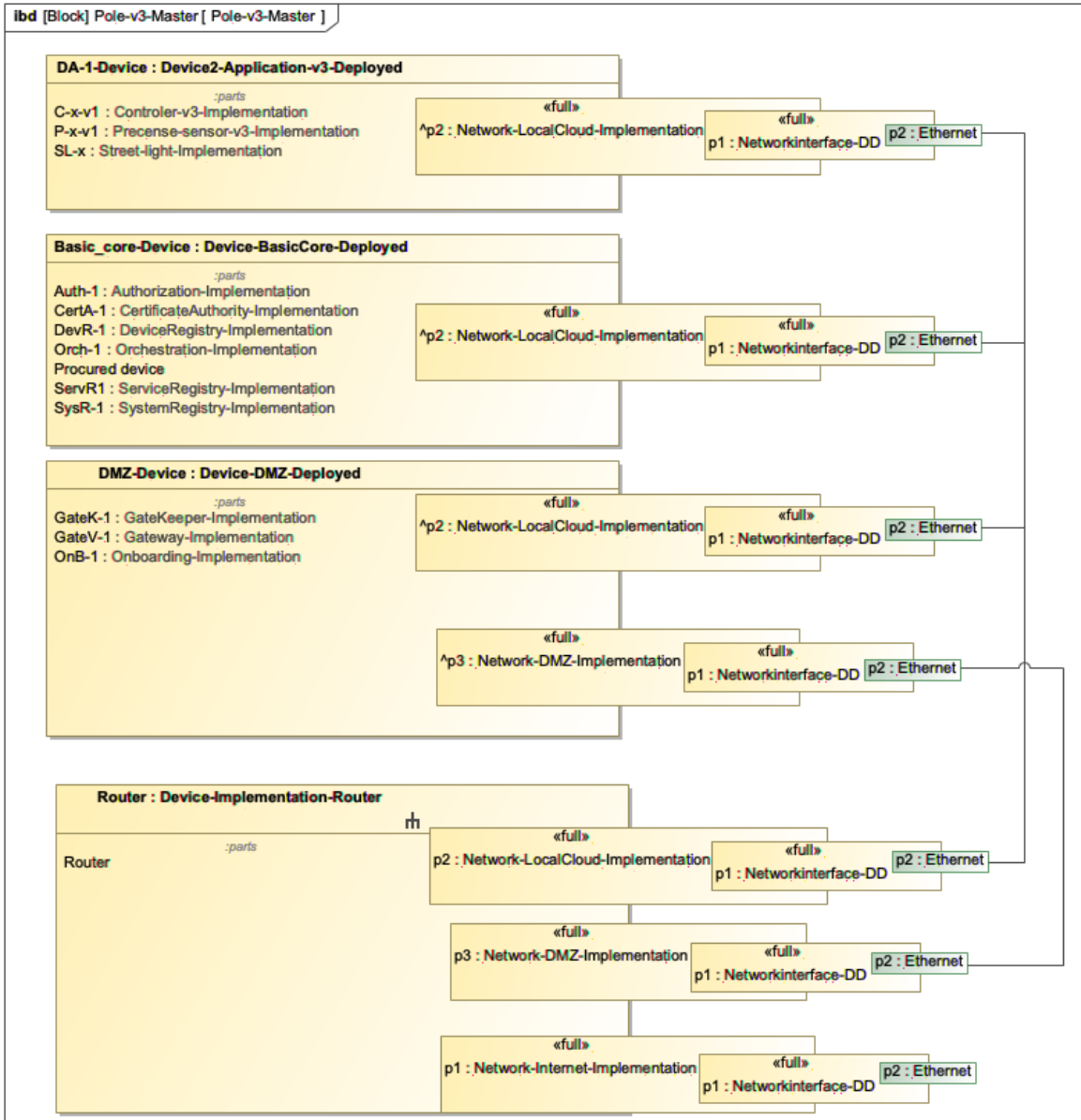
SoS architecture and engineering in SysML



Deployment engineering

ARROWHEAD TOOLS

SoS/Local cloud implementation engineering



Network deployment

#	Name	Role	Role
1	Etnernet connections Dev 1	inout p2 : Ethernet	inout p2 : Ethernet
2	Etnernet connections Dev 2	inout p2 : Ethernet	inout p2 : Ethernet
3	Etnernet connections Dev 3	inout p2 : Ethernet	inout p2 : Ethernet
4	Etnernet connections Dev 4	inout p2 : Ethernet	inout p2 : Ethernet
5	Etnernet connections Router	inout p2 : Ethernet	inout p2 : Ethernet

ToDos

ID of instances to be made according to standards

Automatic naming of instances based of standards

Applicable standards

ISO 15926

ISO 10303

ISO 19650 - BIM v5

ARROWHEAD TOOLS

Deployment of policys

Orchestration policys

- PlantDescription system
- Management system

} Orchestration system

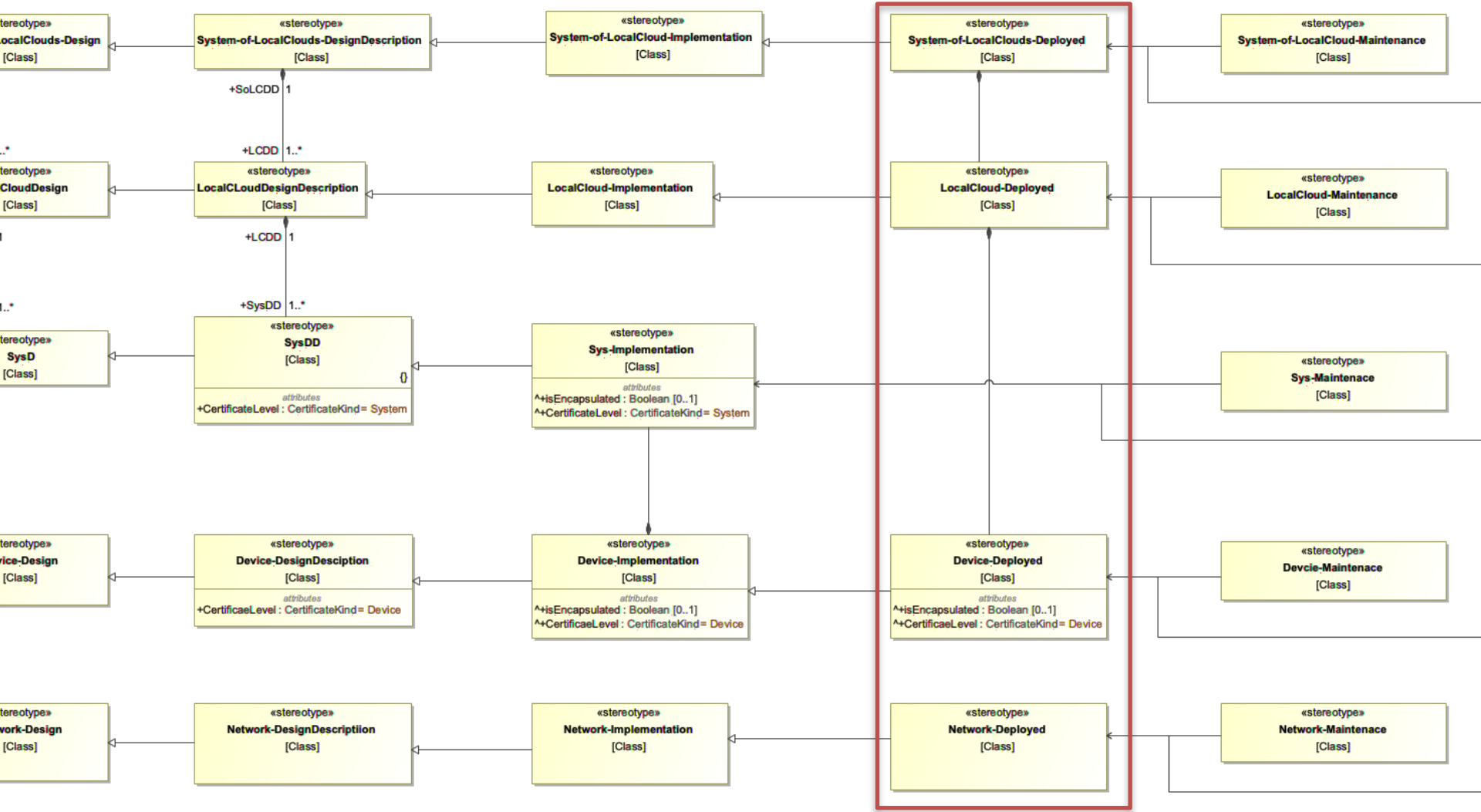
Authorisation rules

- PlantDescription system
- Management system

} Authorisation system

ARROWHEAD TOOLS

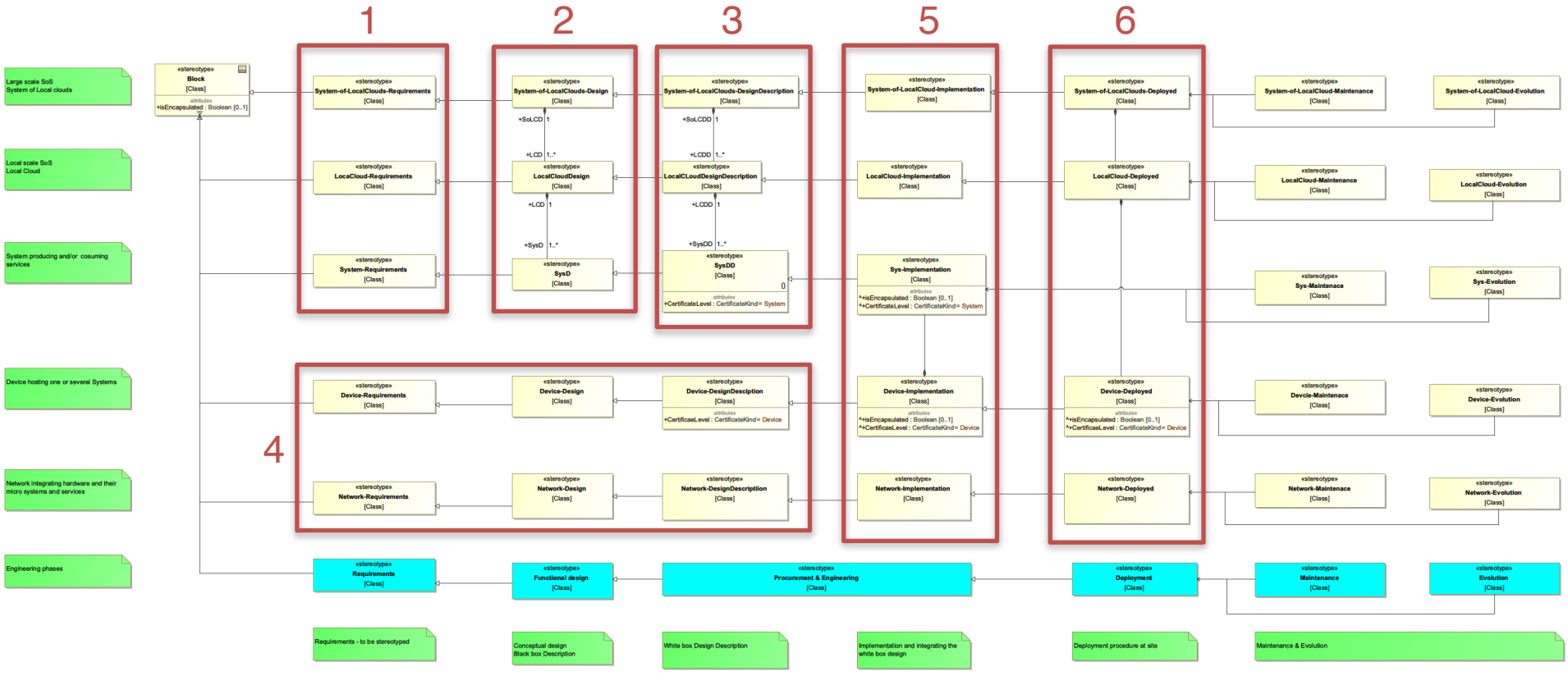
SoS architecture and engineering in SysML



SoS SOA engineering path

ARROWHEAD
TOOLS

SoS architecture and engineering in SysML



ARROWHEAD TOOLS

Availability

Github

https://github.com/jerkerdelsing/SysML_profile_library

will move to GitHub.com/eclipse-arrowhead/profile-library-SysML soon

Owncloud

https://atmospheres.research.ltu.se/owncloud/index.php/apps/files/?dir=/Arrowhead_Tools/WP3%20Arrowhead%20Framework/Task%203.1/Work/SysML_Arrowhead_profile/Profile/JD/Distribution/SysML_profile_library&fileid=1940133

ARROWHEAD TOOLS

Next step

Next step

Move from here to Docker containers for deployment to

Selected HW and OS

- Server
  - HP …, Linux - Ubuntu 20.10
- Desktop computer,
  - Dell…, Windows 10.xx,
  - MacBookPro, OSX 11.2.1
- Embedded system e.g.
  - Raspberry PI, Ubuntu PI4

ARROWHEAD TOOLS

Next step

Models of support core systems to be releases

Models of adaptor systems

- OPC-UA -> Arrowhead - Aparajita, An
- Modbus TCP -> Arrowhead, TWT
- Z-wave -> Salman

Models of libraries

- Python - Jacob
- Kalix - Emanuel
- C++ - Szvetlin

ARROWHEAD TOOLS

Next step

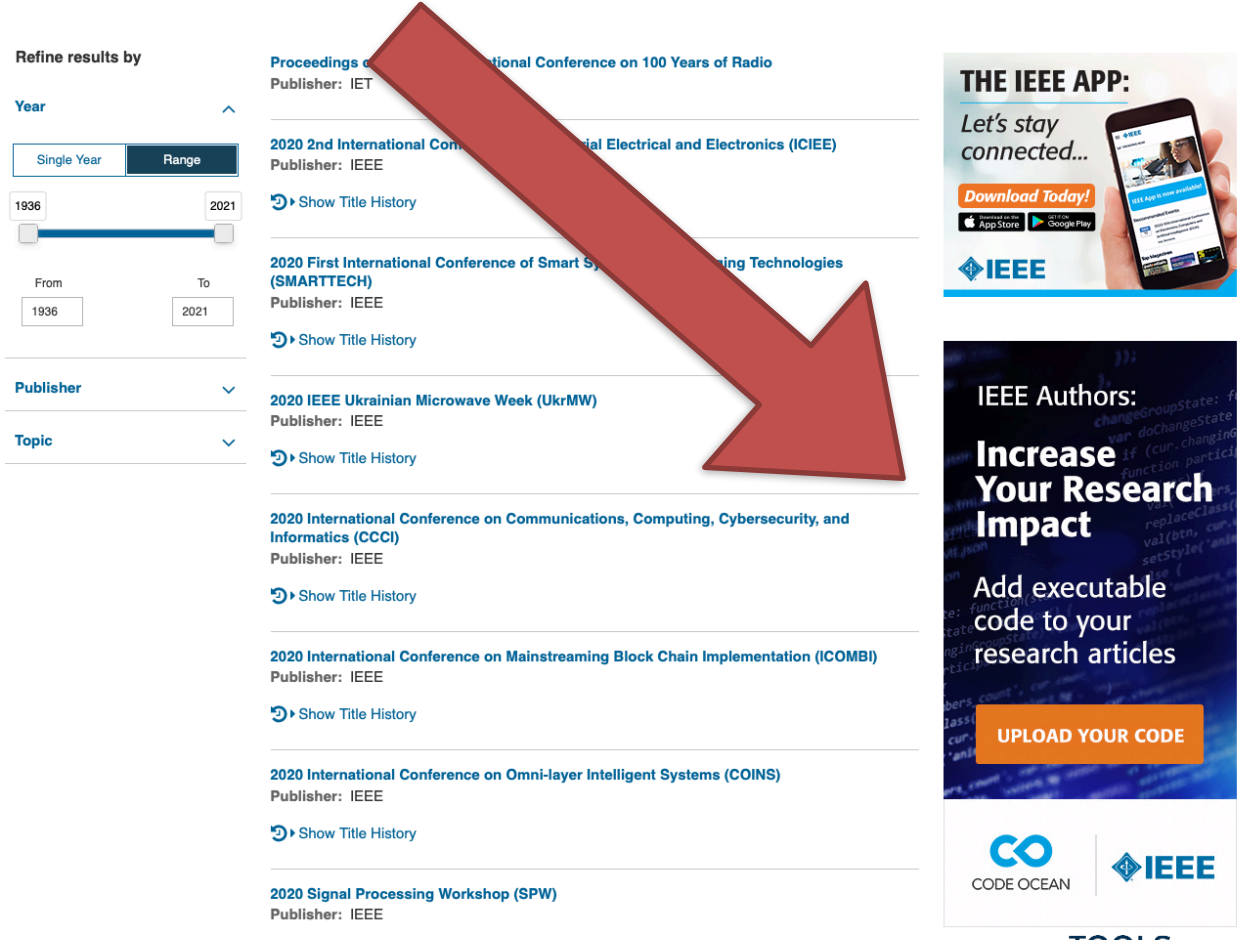
Modelling of other systems

- I need documentation
  - SysD
  - SD
  - IDD
  - code pointer
    - package pointer is good
- You get
  - Feedback on documentation
  - Possibilities to validated your results in a larger environment
  - Enables your results to be used by others

ARROWHEAD TOOLS

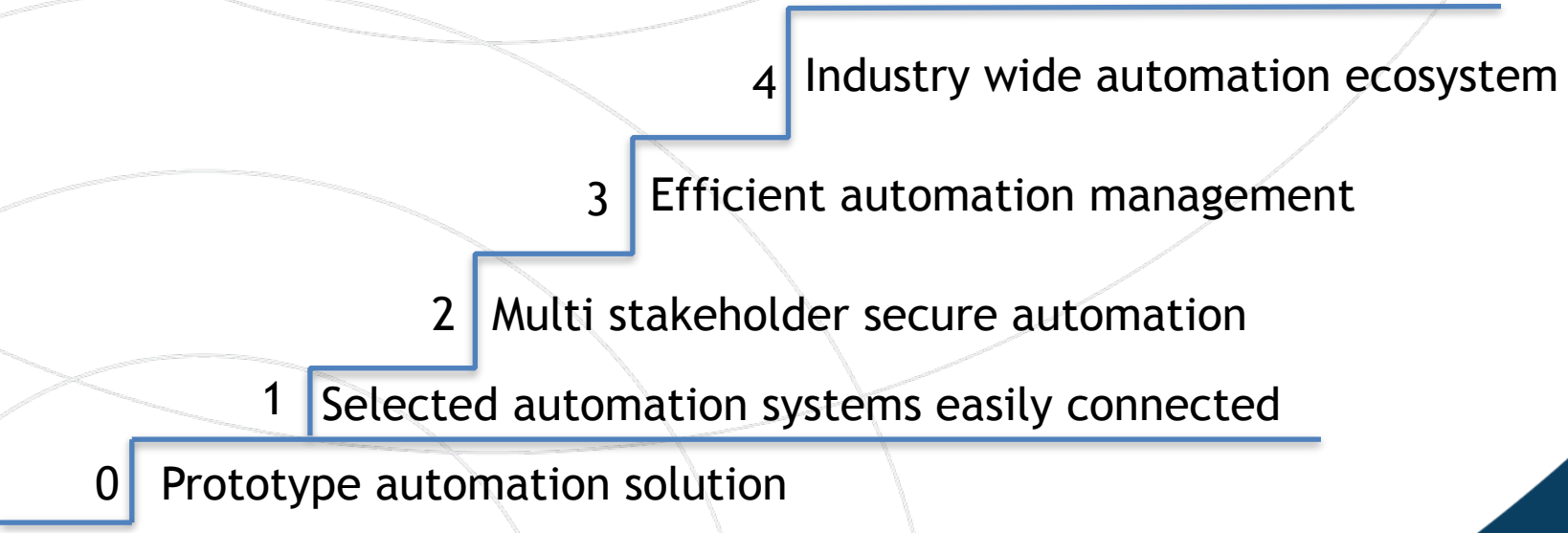
Increase impact of your results

Screen shot from IEEE Explorer today



Automation engineer view

Functionality stair



ARROWHEAD TOOLS

Next step

Training - artefacts

- Eclipse Arrowhead “Hello World” example
- Utilising the ServiceRegistry
- Utilising the Orchestration system
- Utilising the Authorisation system
- Building your own application
- Using support core systems
- Multiple local clouds
- Security
- Run-time engineering
- Solution modelling and engineering

ARROWHEAD TOOLS

Comments! Questions?

jerker.delsing@ltu.se

ARROWHEAD TOOLS