

# SysML tooling for design and engineering of System of Systems

Prof. Jerker Delsing

Luleå University of Technology

Sweden



# What is SysML

The **Systems Modeling Language (SysML)**<sup>[1]</sup> 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.<sup>[2]</sup> 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.

# 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

# System of Systems modelling

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

# 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

# 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.

**Emergent behavior** appears in any system, and in many systems this is deliberately and intentionally designed in, and tested. 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.

**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

# SysML modelling basics

Requirement diagram/table

Use case diagrams

Activity diagram

Block definition diagrams

Internal block diagrams

Parametric diagram

State machine diagram

Sequence diagrams

...

...

# 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

# SOA support, cont

Library

Eclipse Arrowhead core systems

Templates for

Local clouds

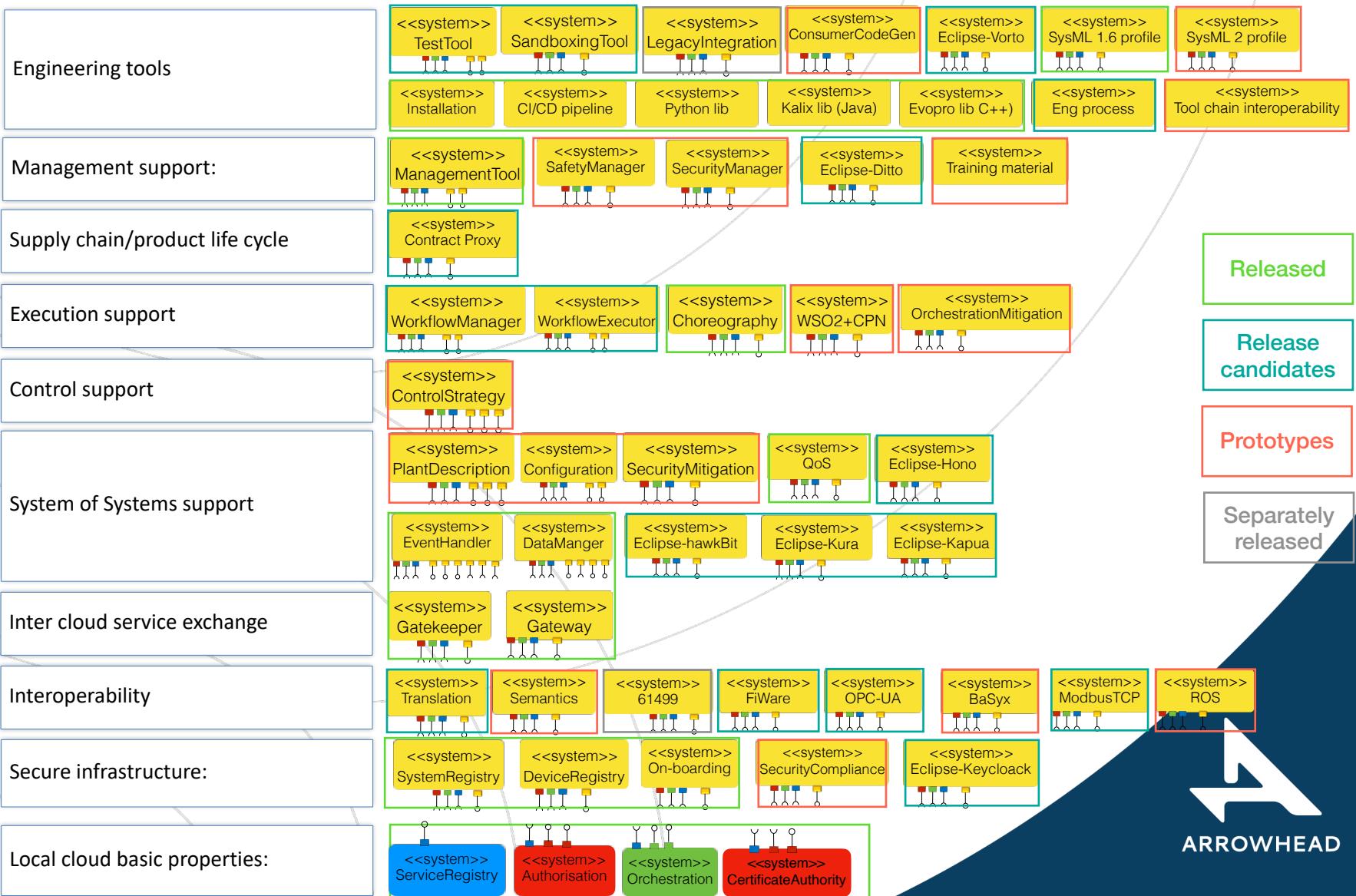
System of Local clouds

Generic application systems

Devices

Network

# Arrowhead v4.3.0



Released

Release candidates

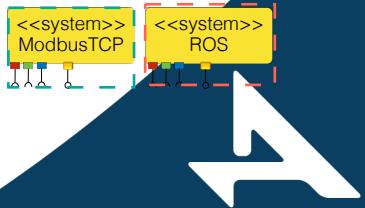
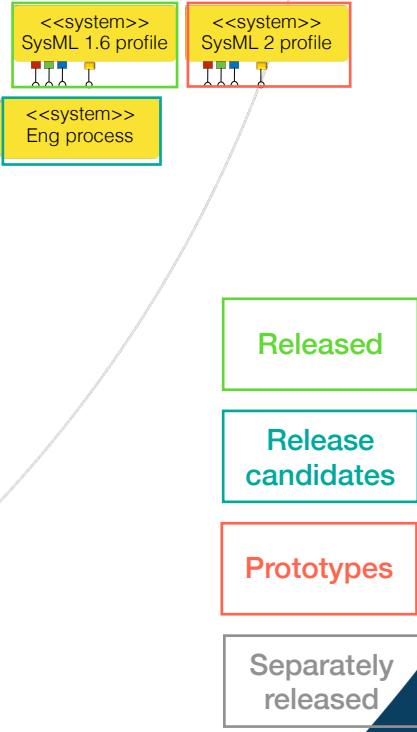
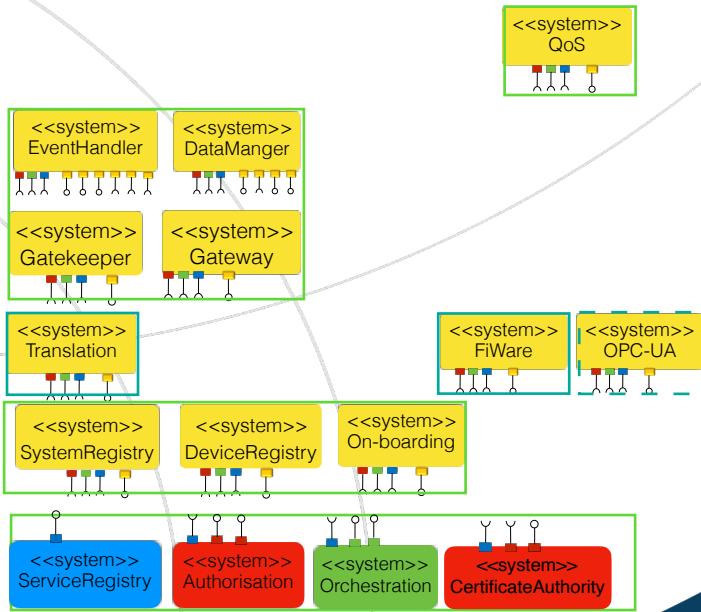
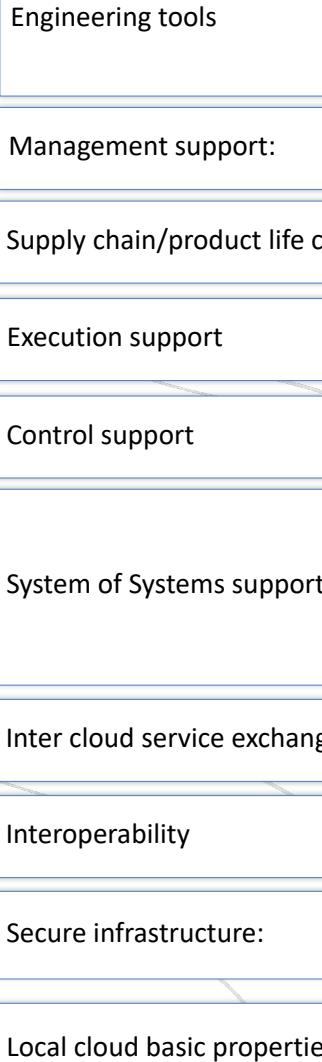
Prototypes

Separately released



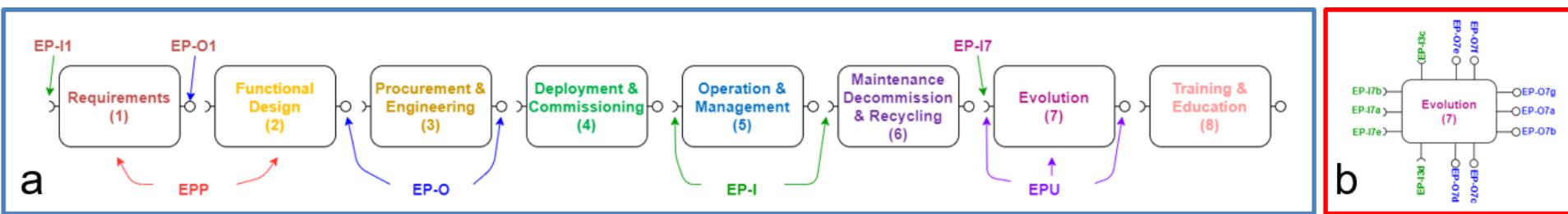
ARROWHEAD

# Arrowhead v4.3.0



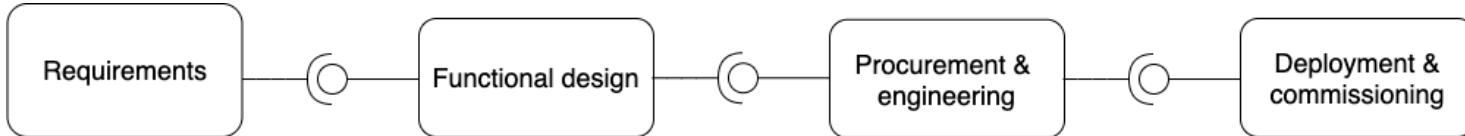
ARROWHEAD

# Engineering process



# Eclipse Arrowhead engineering

Engineering process  
IEC 81346



## 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,

SoLCD, LCD,  
SysD, SD

## Procurement & engineering

### Procurement of:

- Application hardware, OS, Router,
- Installing OS

### White box engineering of:

- Application systems and services code
- Orchestration and security polycys
- Installing core and application systems to procured HW
- Configuration of network,

LCDD, SysDD, IDD

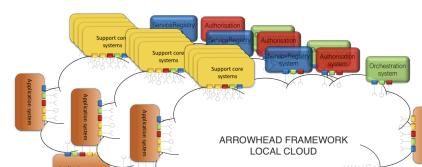
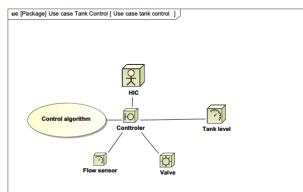
## Deployment & commissioning

### Deployment of:

- HW with core and application systems in plant,
- Orchestration policycs
- Security policycs

### Commissioning of:

- Local cloud functionality
- System of local cloud functionality



## Core system selection

## Devices and network



## Physical deployment at site

- Devices
- Routers
- Power supply
- Network connection

## OS



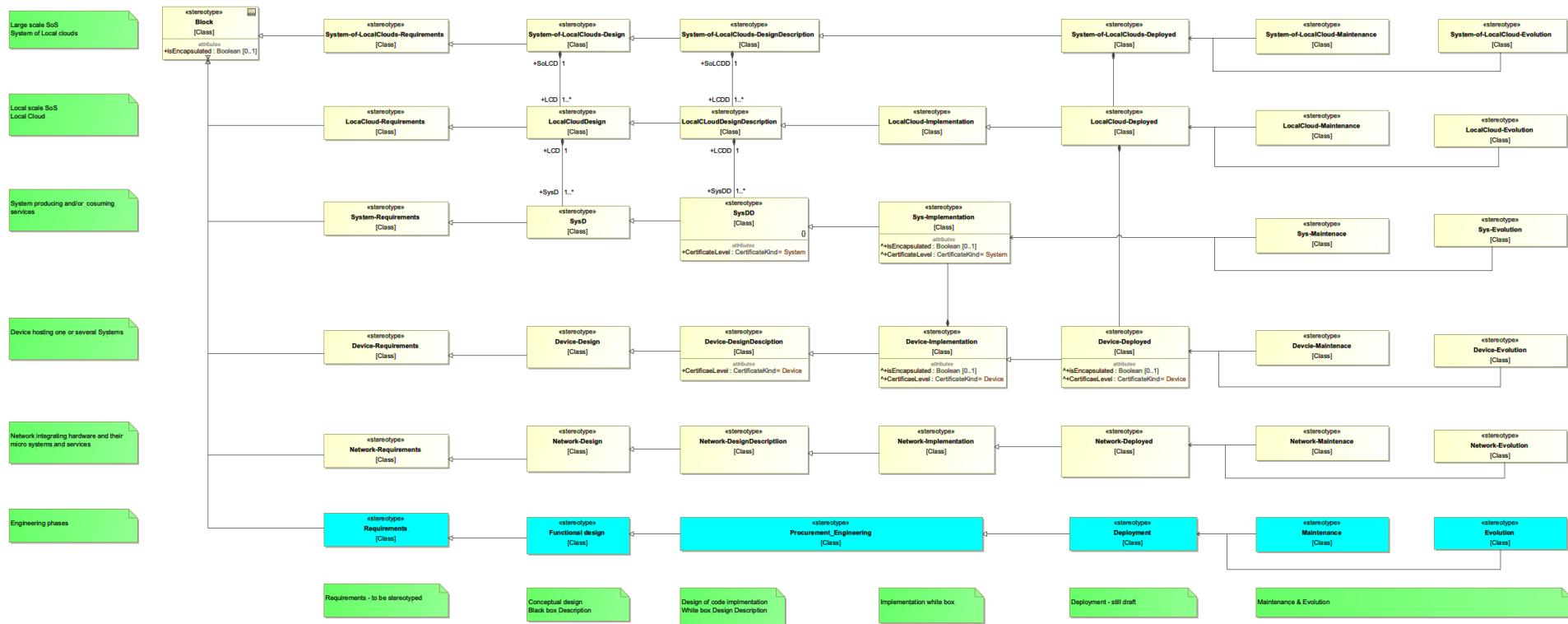
## Network



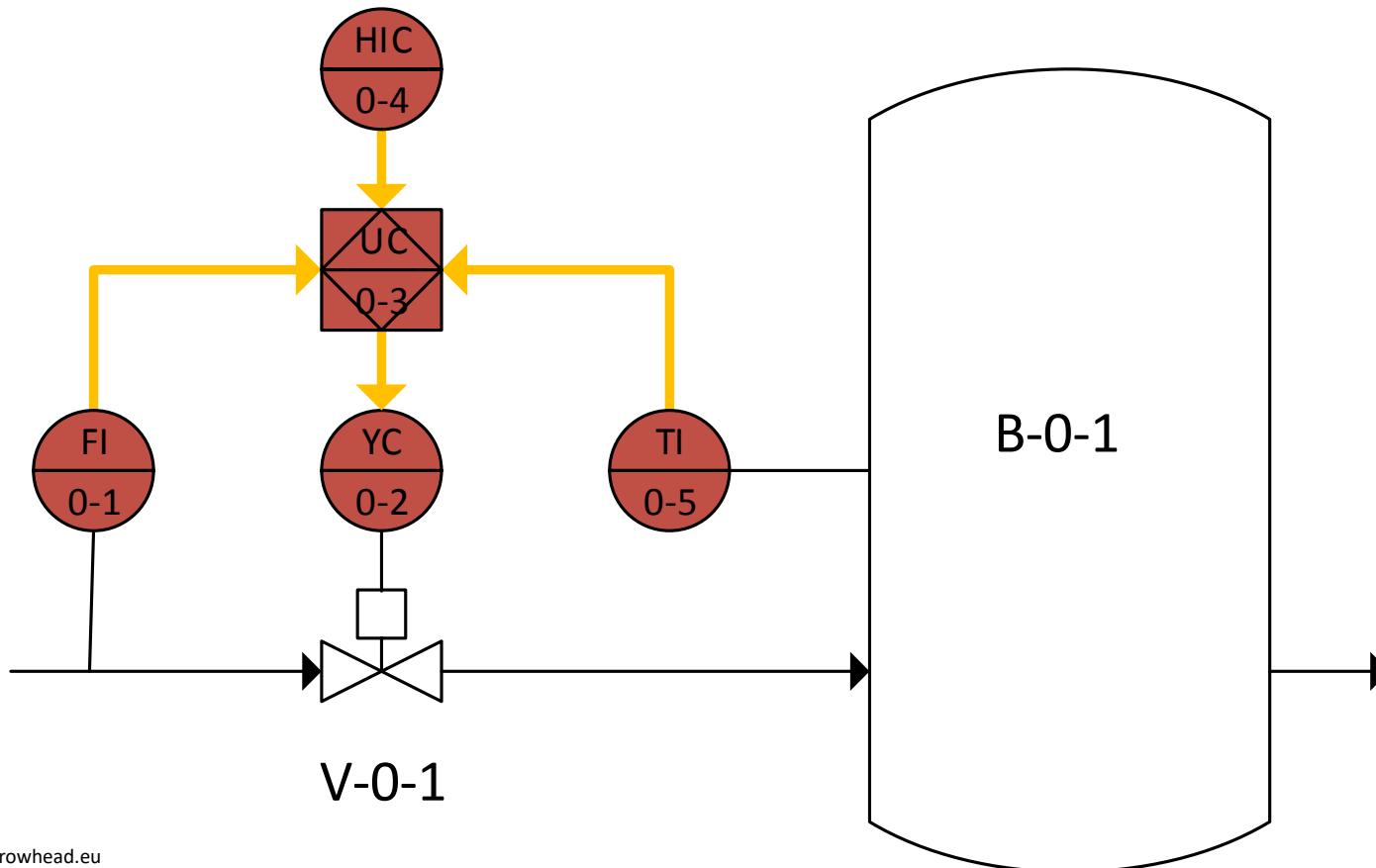
## Application system & service design



# SoS architecture and engineering in SysML

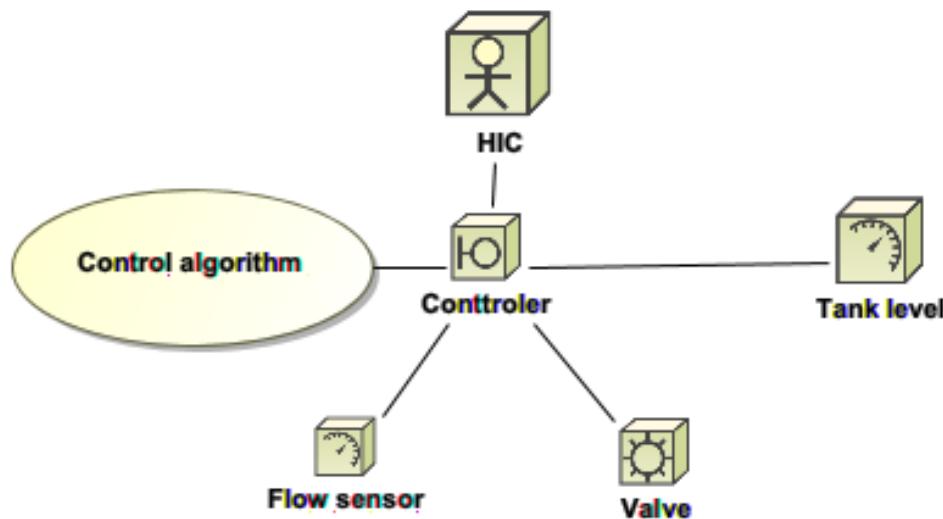


# Lets start with a use case



# Use case diagram

uc [Package] Use case Tank Control [ Use case tank control ]



# Requirements

req [Package] Tank control Local cloud [ Tank control Local cloud ]

«requirement»

Id = "17"

Text = "Tank level control function. Based on level sensor, flow measurement and "

«functionalRequirement»

Id = "18"

Text = "Level measurement accuracy: +/-1cm

Flow measurement accuracy of actual flow: 1%

Valve flow control: linear

Tank level max: 90%

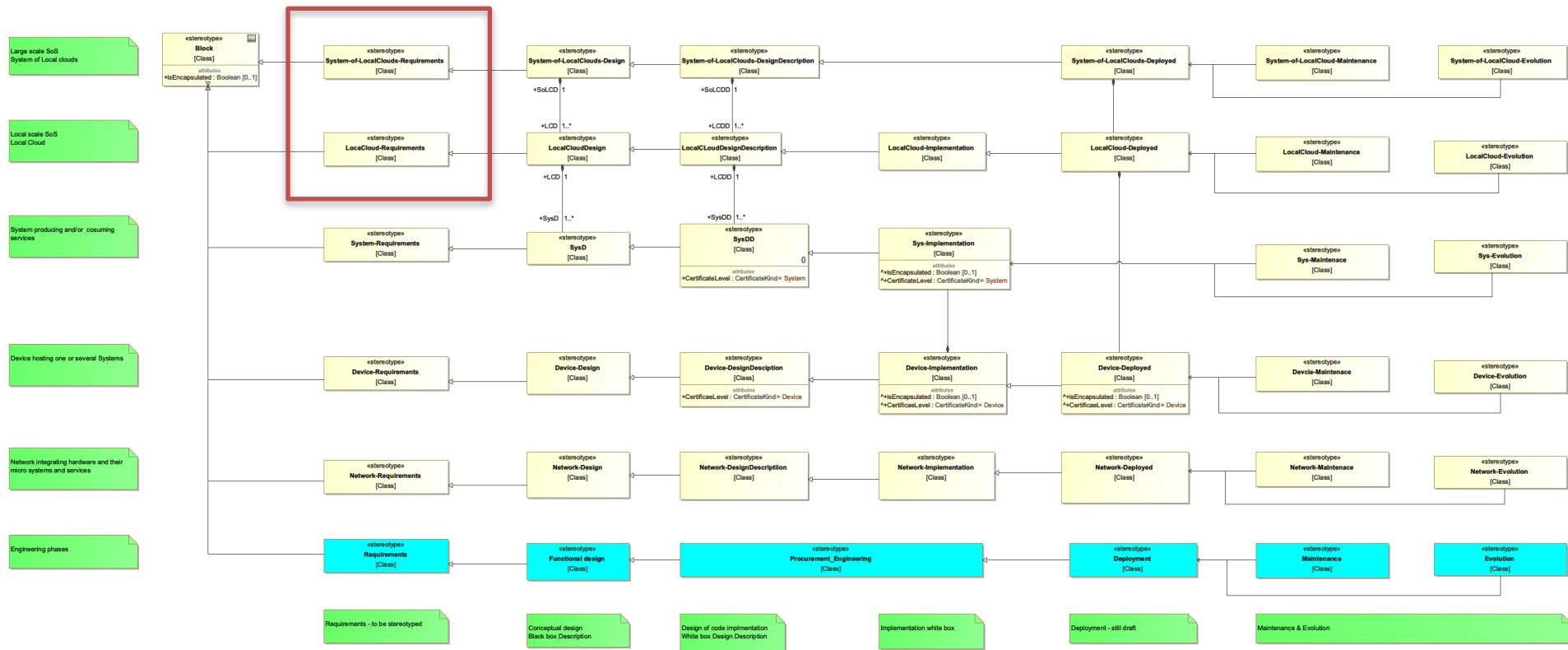
Tank level min: 10%"

«performanceRequirement»

Id = "19"

Text = "Controller cycle time: 1s"

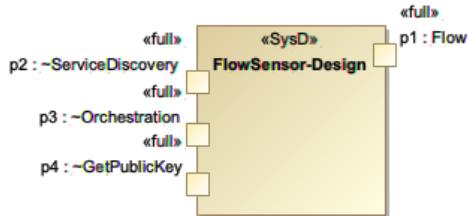
# SoS architecture and engineering in SysML



# Functional system and service design

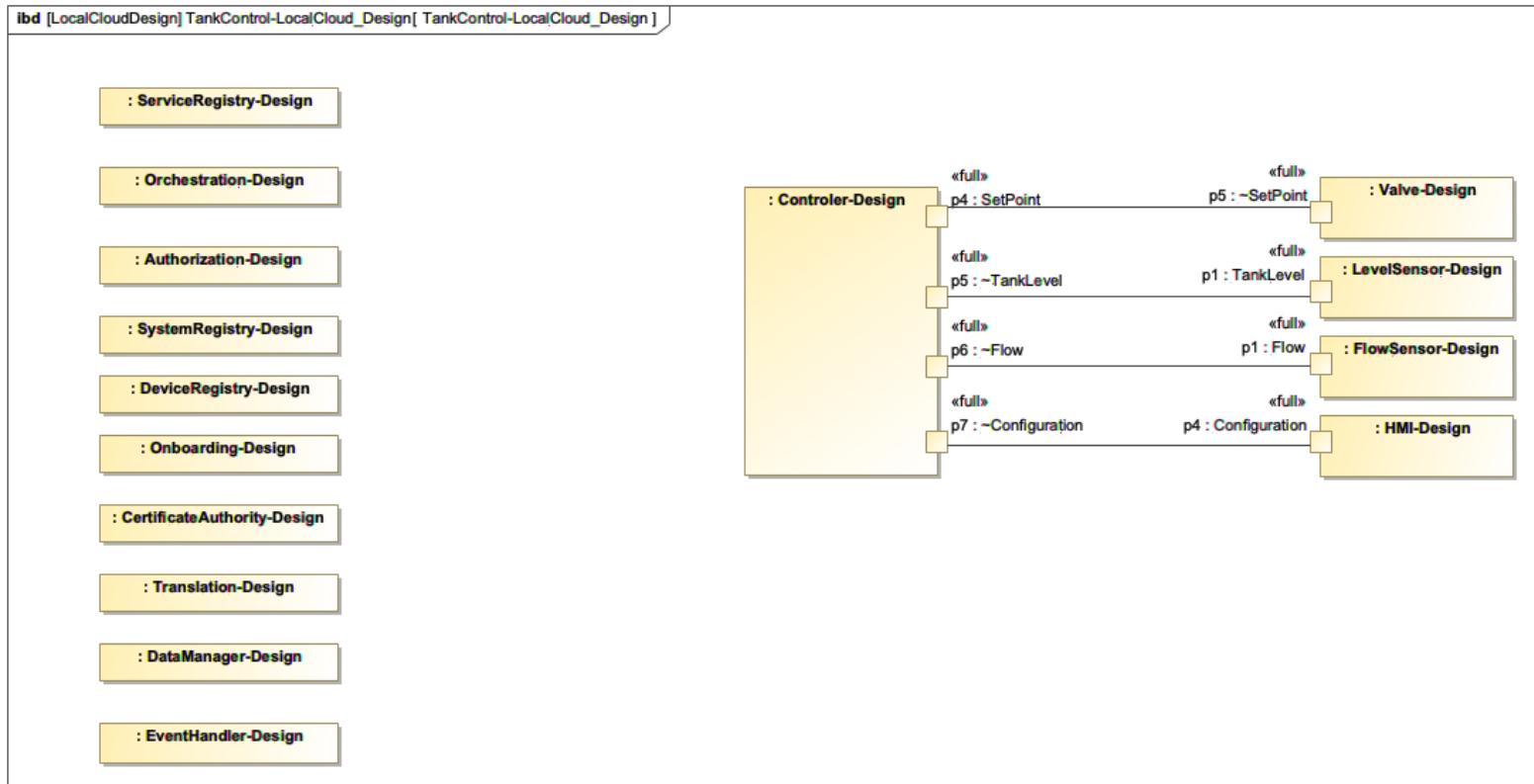
Micro-systems

Micro-services

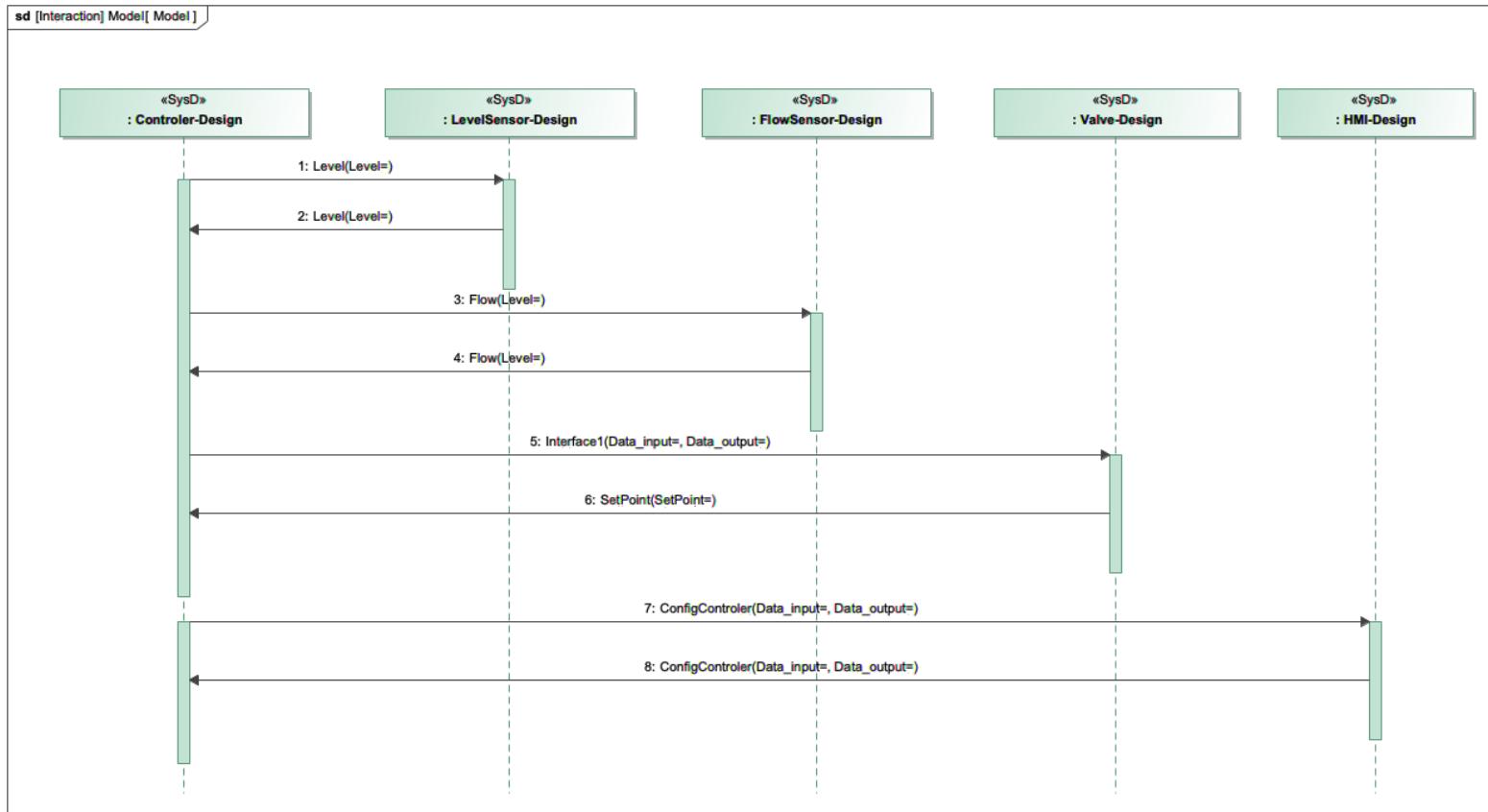


# Functional SoS design - black box

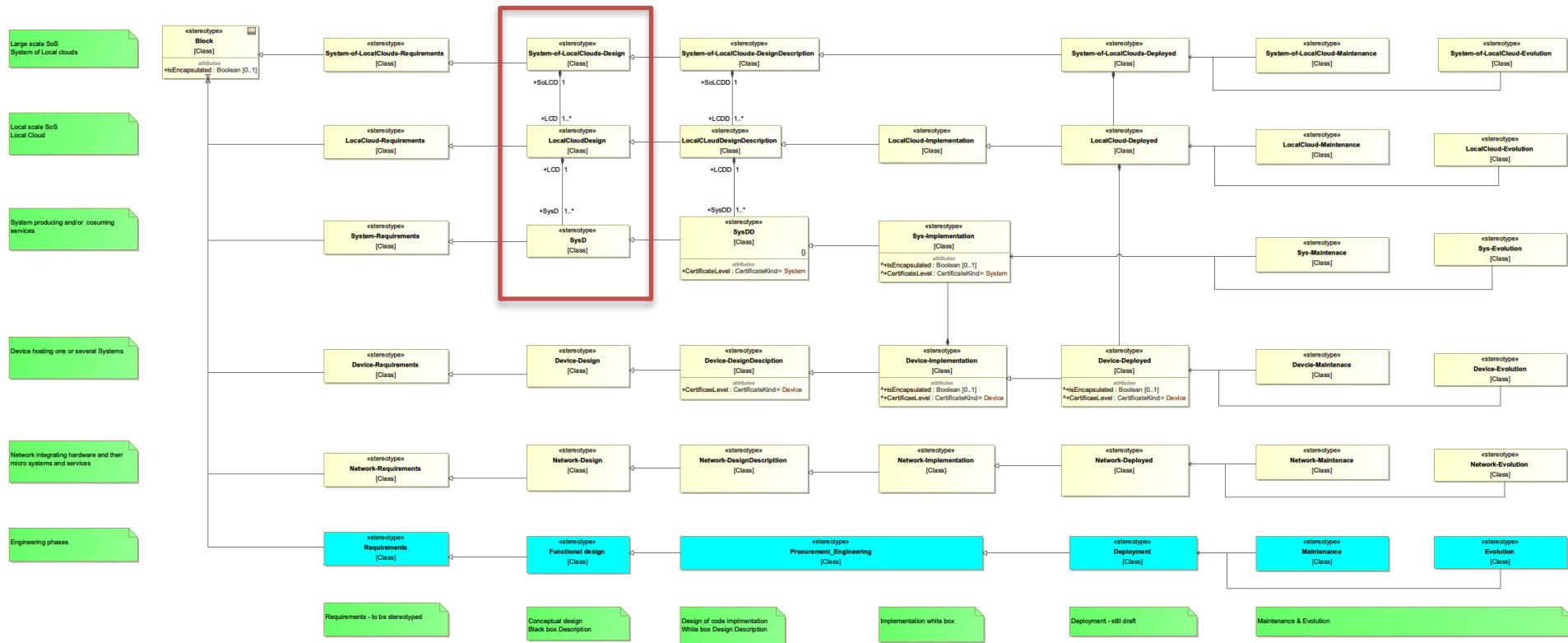
## Local cloud functional orchestration



# Service exchange functionality



# SoS architecture and engineering in SysML



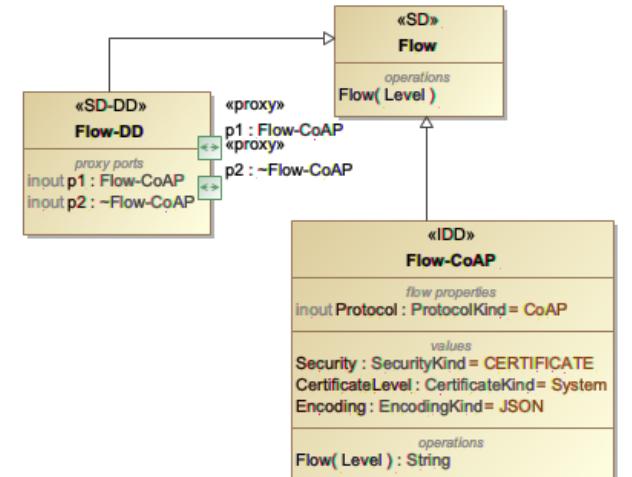
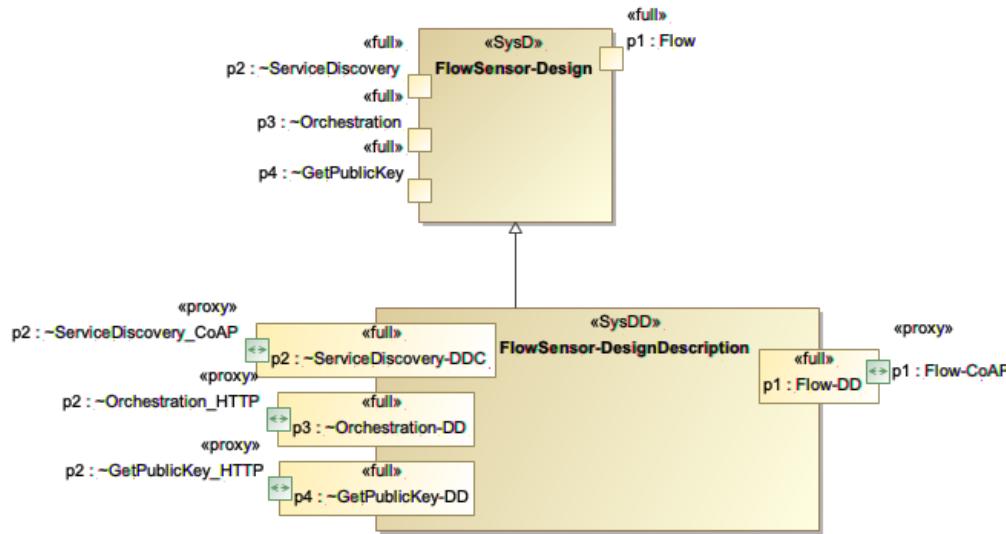
# White box engineering

SysDD and IDD

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

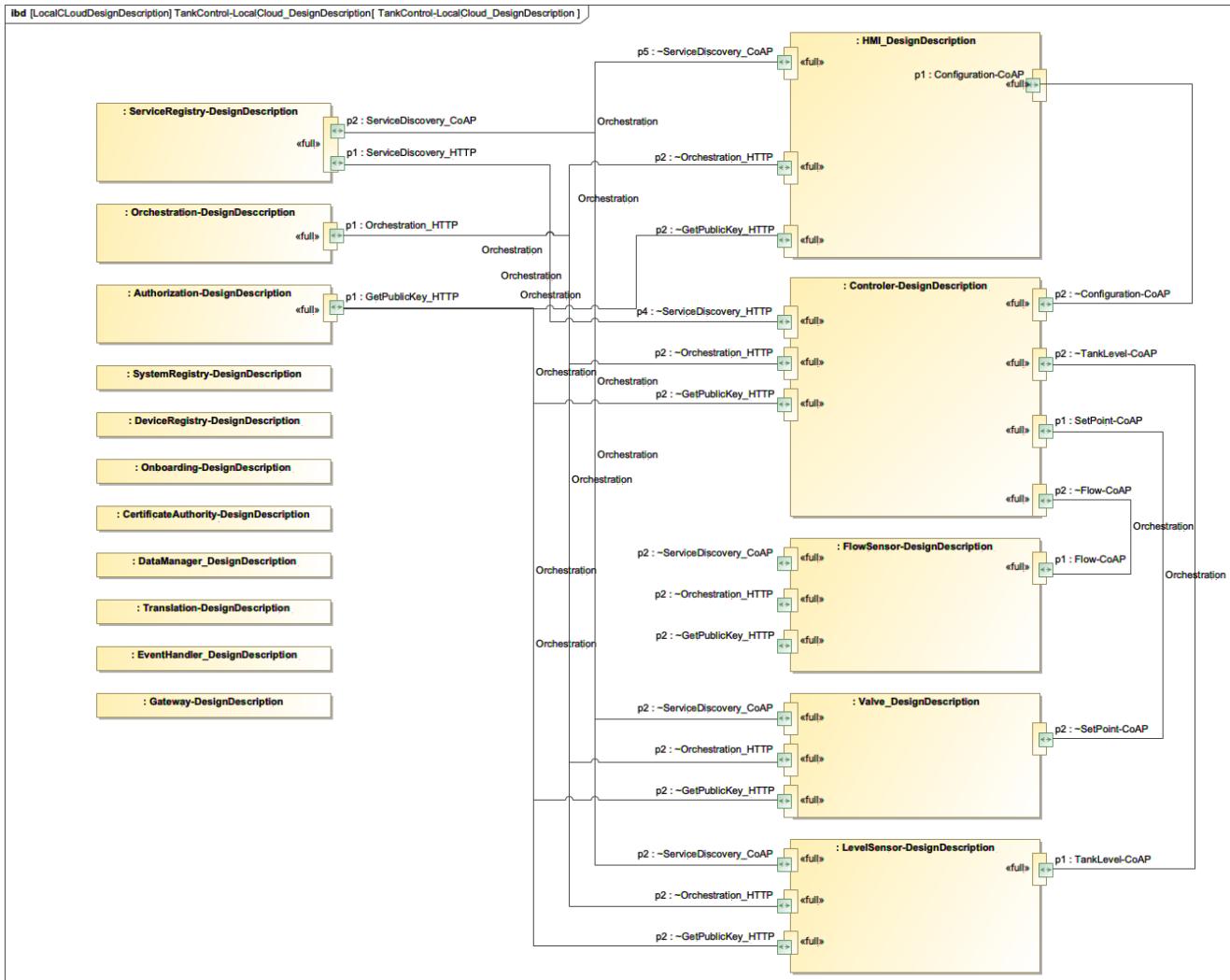
## Micro-systems

## Micro-services



# Functional LC design description - white box

Local cloud functional design description model



# Orchestration policies - rules and conditions

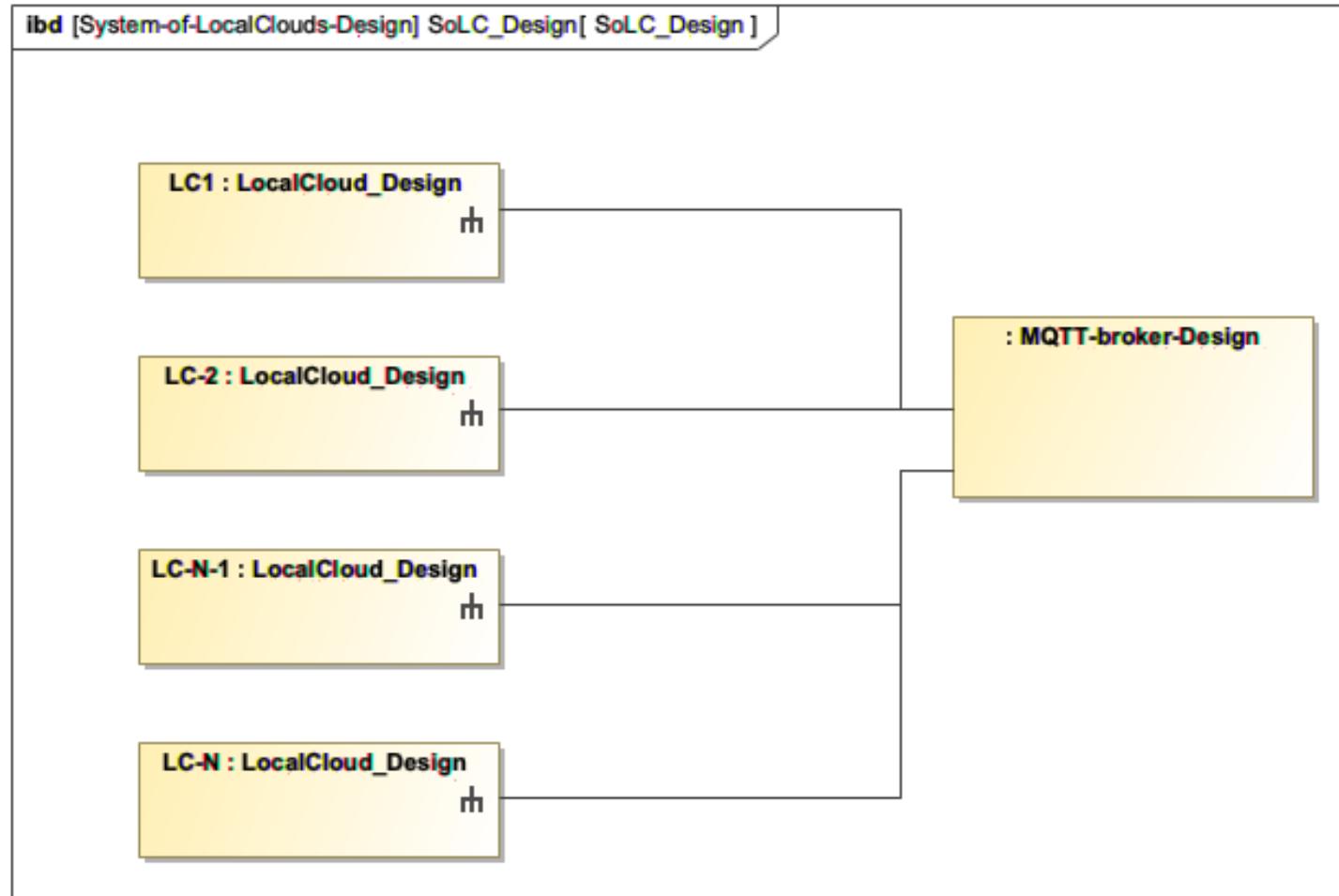
#	Name	Role (Connector End A)	Role (Connector End B)
1	Orchestration	[inout p4 : ~ServiceDiscovery_HTTP]	[inout p1 : ServiceDiscovery_HTTP]
2	Orchestration	[inout p2 : ~Orchestration_HTTP]	[inout p1 : Orchestration_HTTP]
3	Orchestration	[inout p2 : ~GetPublicKey_HTTP]	[inout p1 : GetPublicKey_HTTP]
4	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]

# Security policies - rules and conditions

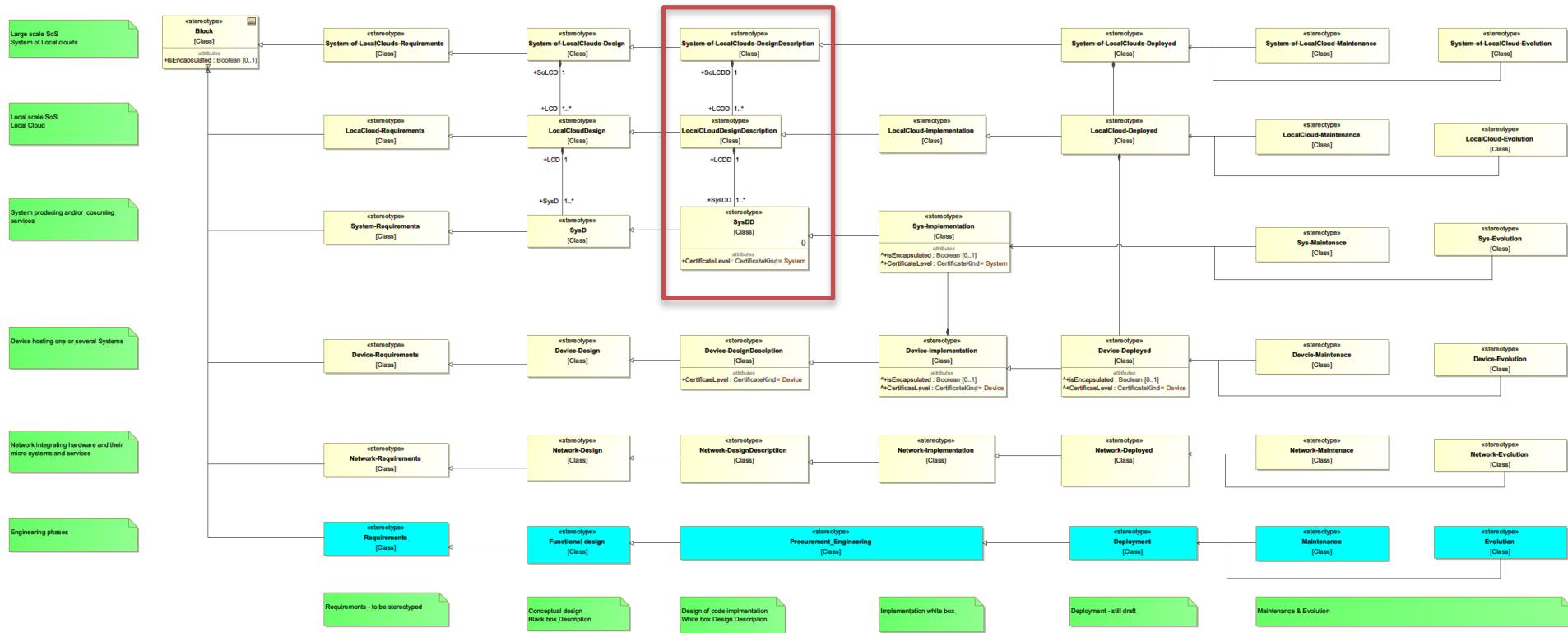
#	Name	Role	Role	Security constrains
1	🔒 Orchestration	[green] inout p4 : ~ServiceDiscovery_HTTP	[green] inout p1 : ServiceDiscovery_HTTP	{ } Security p=authorisation == system certificate: ce...
2	🔒 Orchestration	[green] inout p5 : ~ServiceDiscovery_CoAP	[green] inout p2 : ServiceDiscovery_CoAP	
3	🔒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

# Functional SoLC design

System o local clouds functional design model



# SoS architecture and engineering in SysML



# Implementation

We also need

Devices

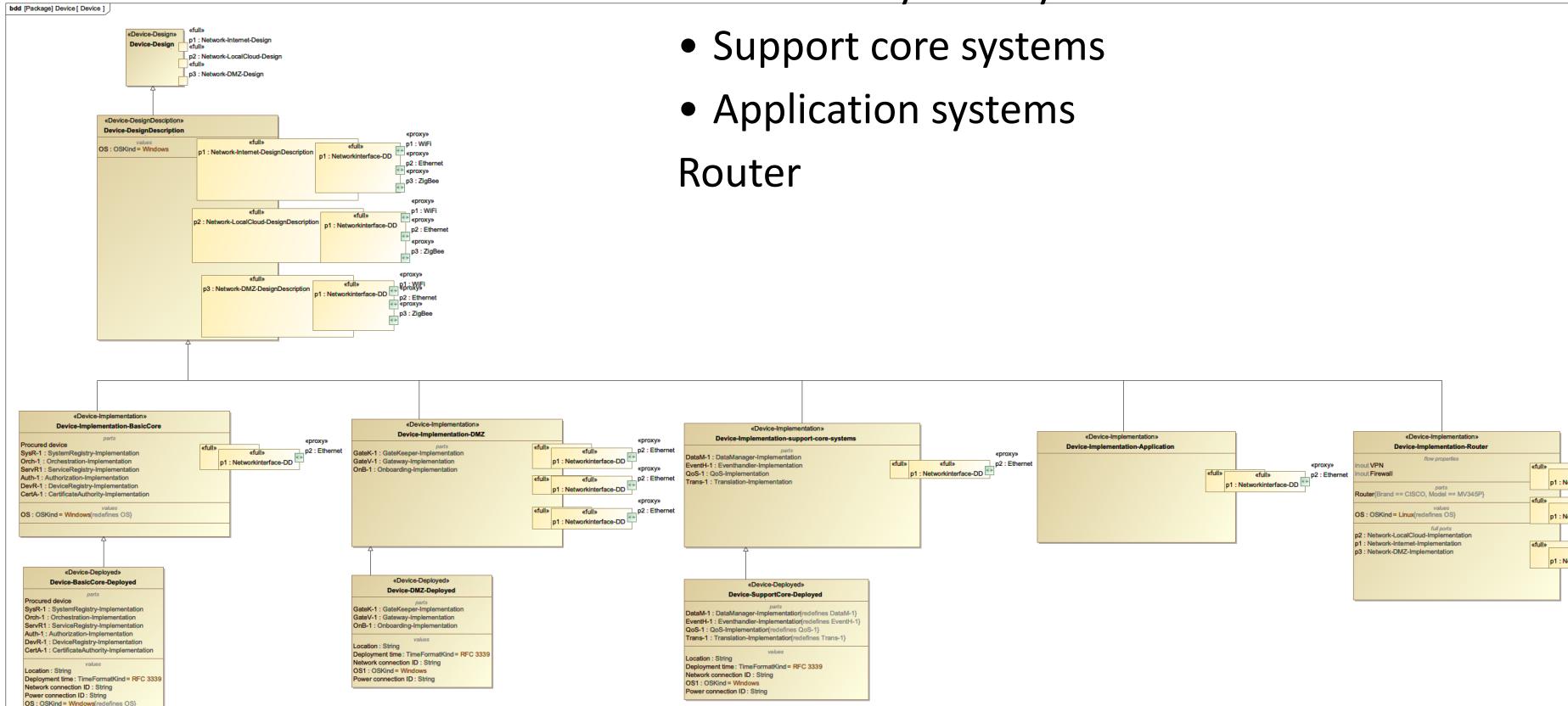
Network

# Device implementation

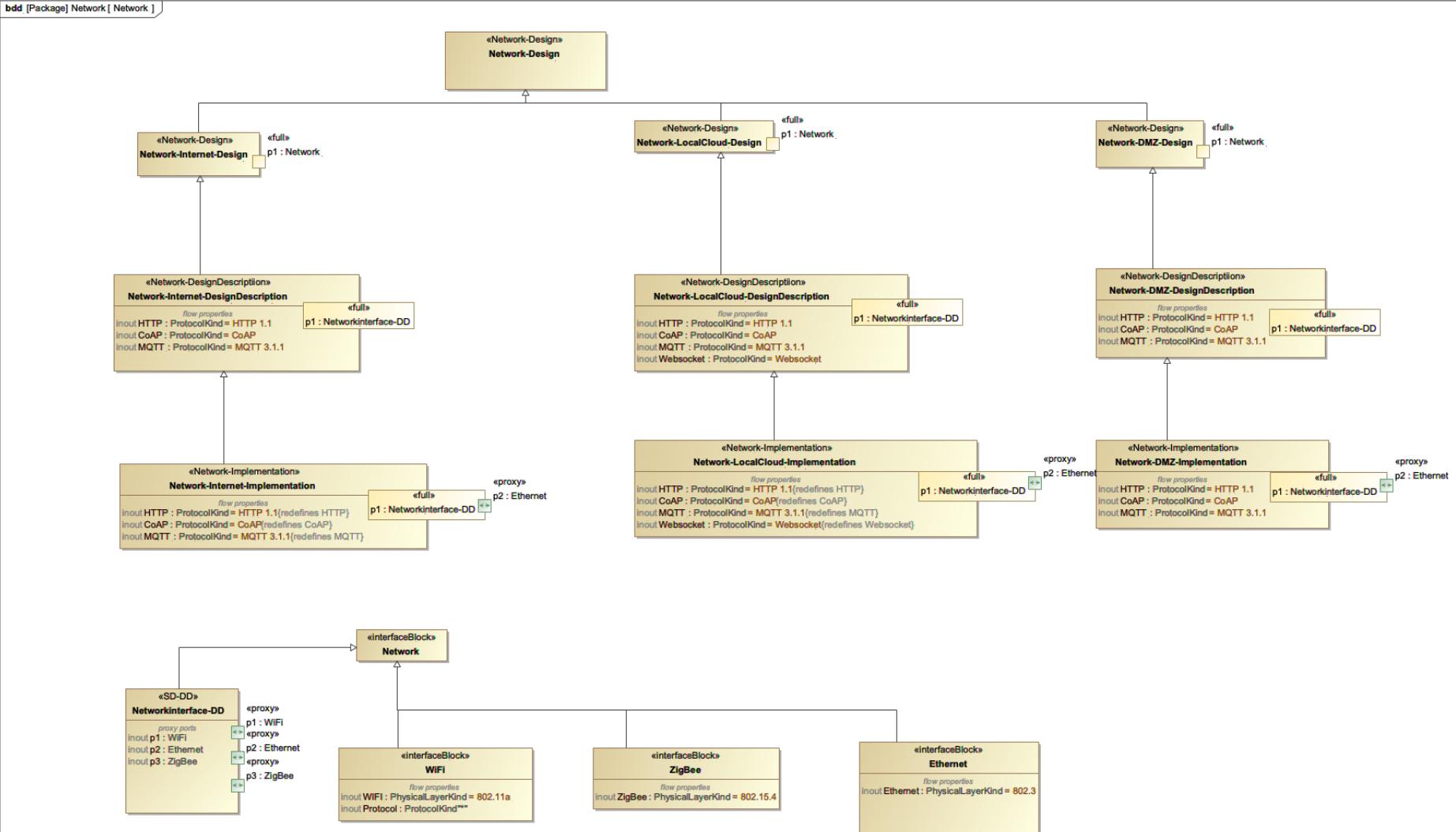
Devices with

- Mandatory core systems
- Support core systems
- Application systems

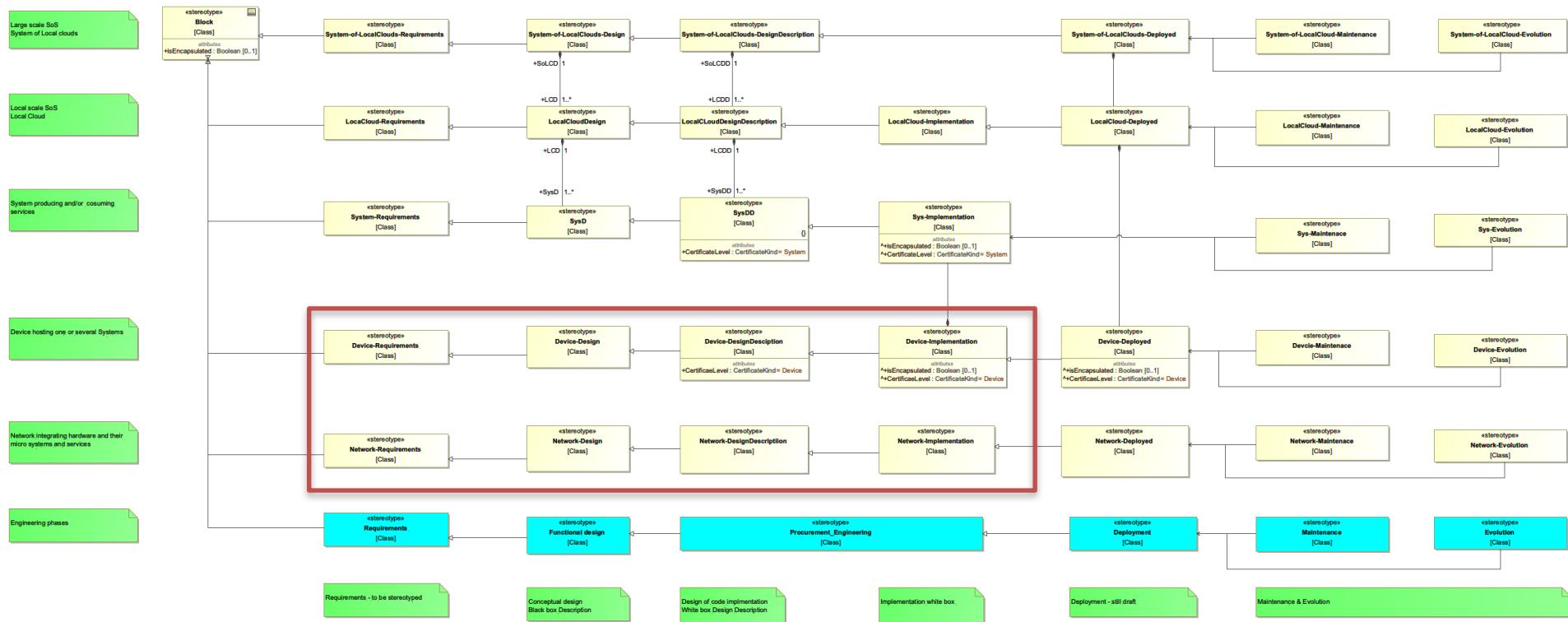
Router



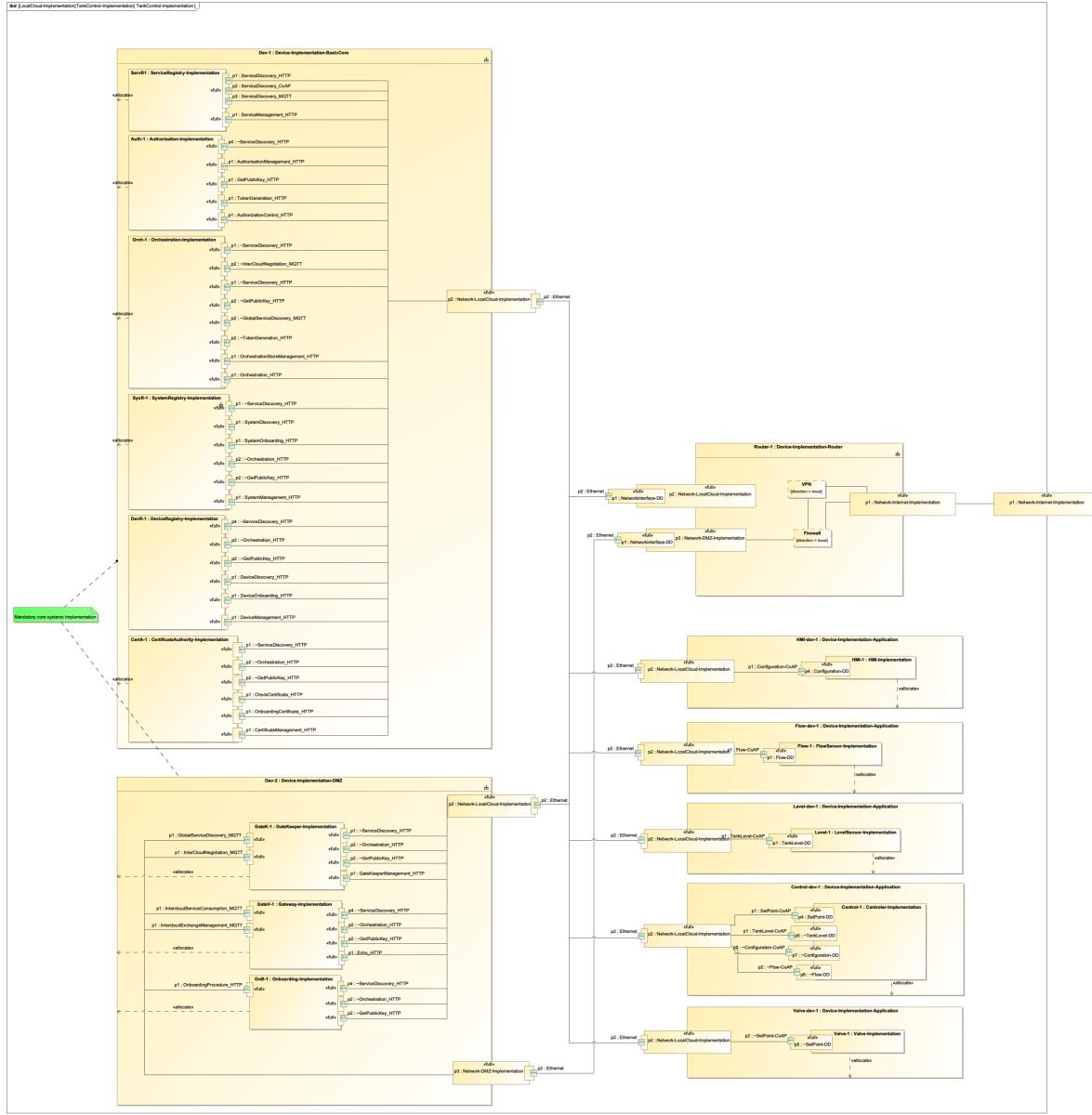
# Network



# SoS architecture and engineering in SysML

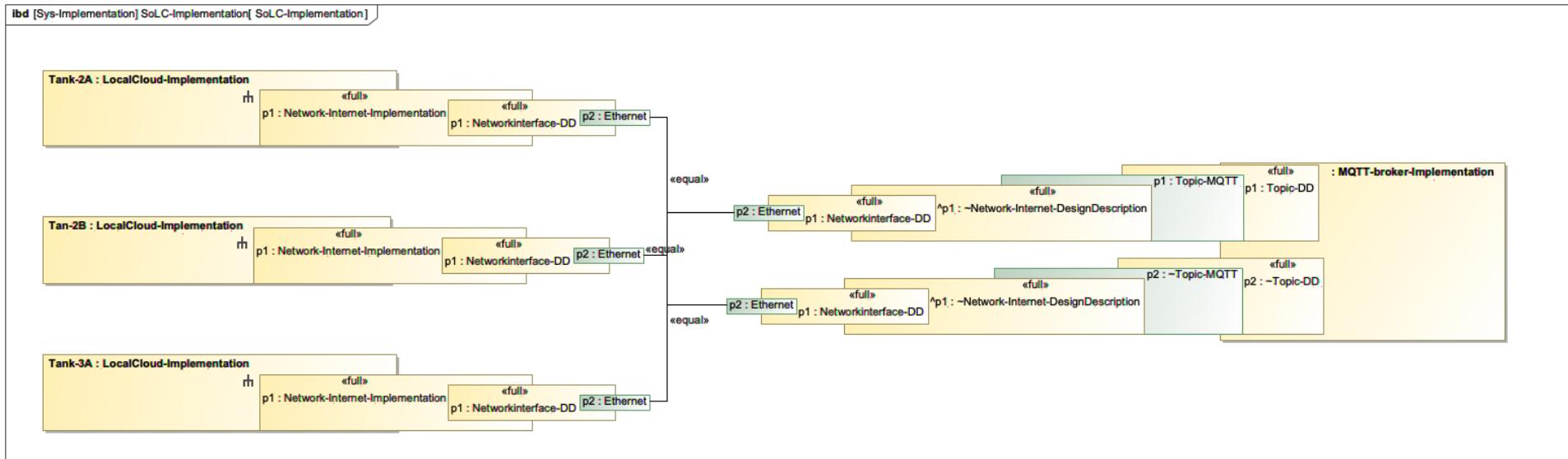


# Functional SoS/ Local cloud implementation engineering



# Functional SoLC implementation engineering

## System o local clouds functional implementation model



# Extraction of code

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

Move from here to Docker containers for deployment to

Selected HW and OS

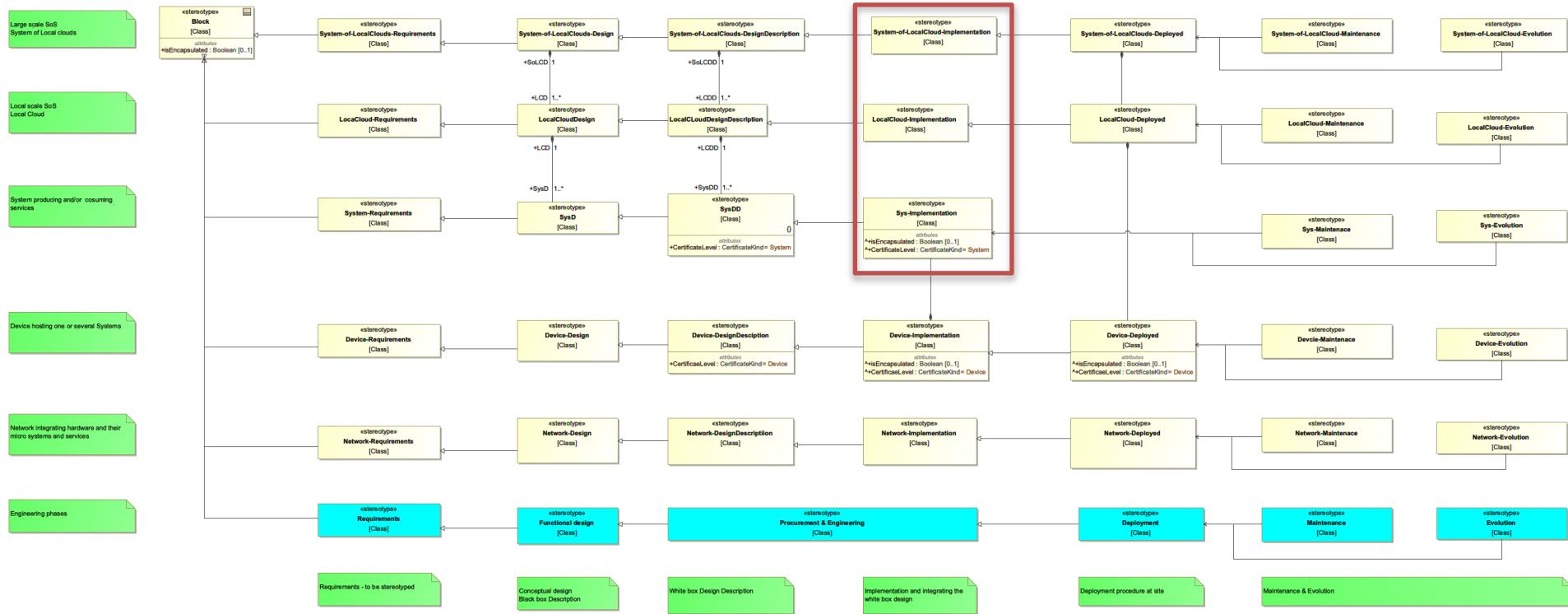
Server - Linux - Ubuntu 20.10

Desktop computer, Windows 10.xx, OSX 11.2.1

Embedded system

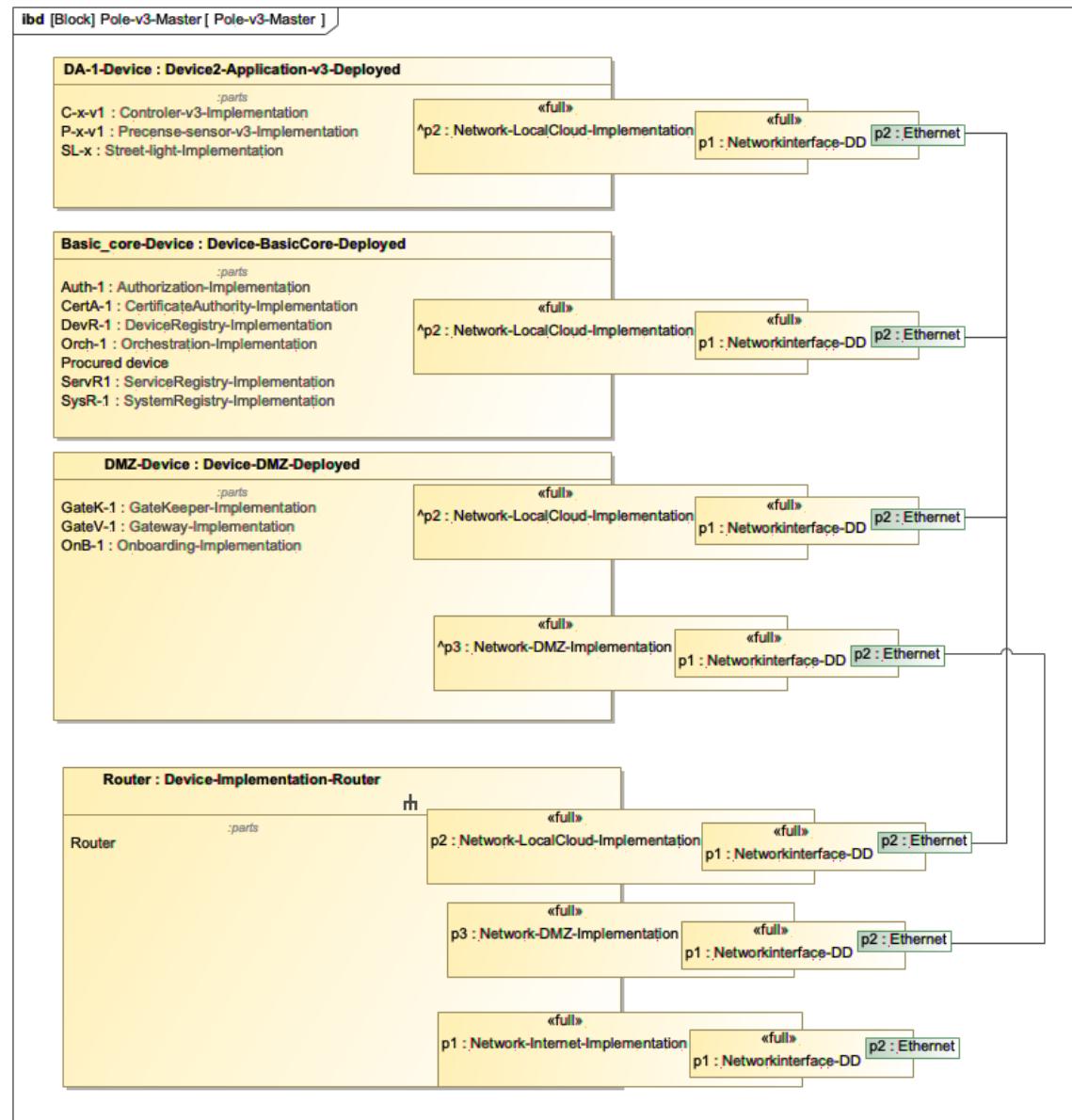
Raspberry PI

# SoS architecture and engineering in SysML



# Deployment engineering

# 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

# Deployment of policys

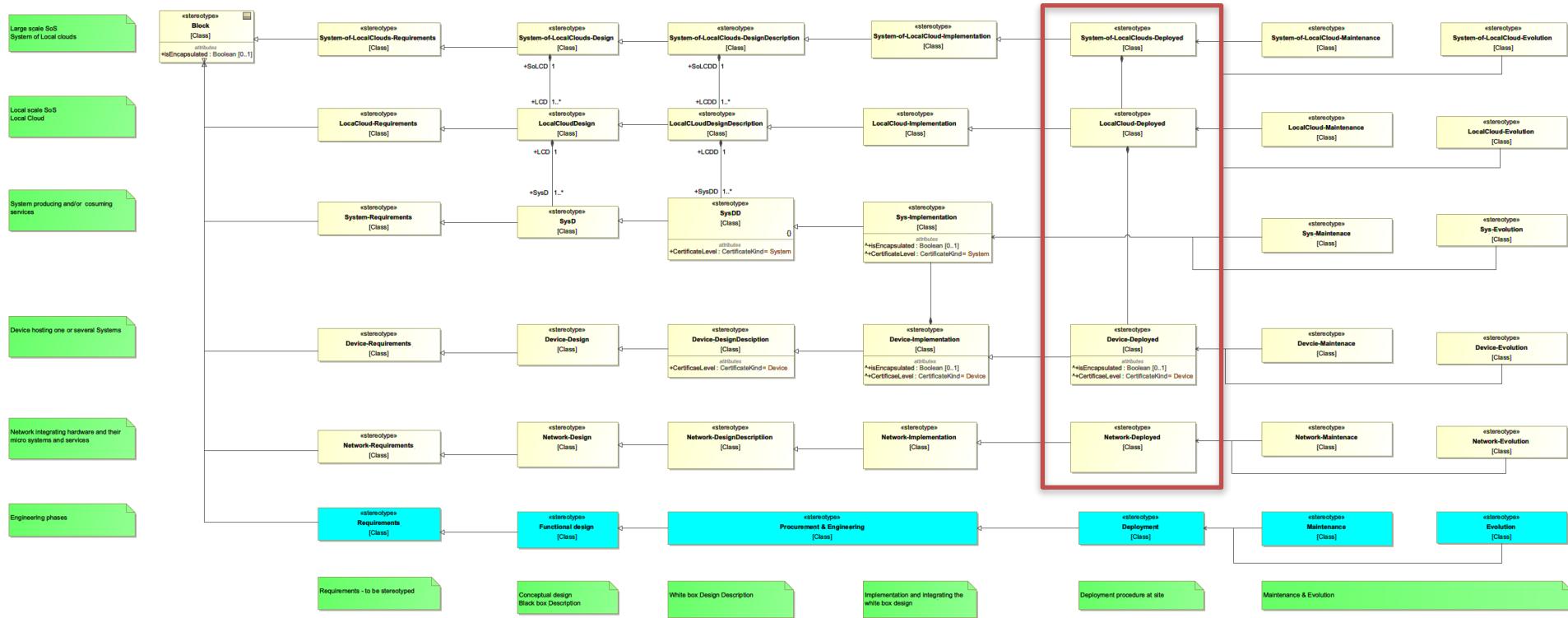
Orchestration policys

PlantDescription system  
Management system } Orchestration system

Authorisation rules

PlantDescription system  
Management system } Authorisation system

# SoS architecture and engineering in SysML



# Availability

Github

[https://github.com/jerkerdelsing/SysML\\_profile\\_library](https://github.com/jerkerdelsing/SysML_profile_library)

will move to GitHub.com/eclipse-arrowhead

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](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)

# 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

# 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

# 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 validate your results in a larger environment

Enables your results to be used by others

# Increase impact of your results

Screen shot from  
IEEE Explorer  
today

Refine results by

Year

Single Year Range

1936 2021

From To

1936 2021

Publisher

Topic

Proceedings of the International Conference on 100 Years of Radio  
Publisher: IET

2020 2nd International Conference on Industrial Electrical and Electronics (ICIEE)  
Publisher: IEEE

Show Title History

2020 First International Conference of Smart Systems and Emerging Technologies (SMARTTECH)  
Publisher: IEEE

Show Title History

2020 IEEE Ukrainian Microwave Week (UkrMW)  
Publisher: IEEE

Show Title History

2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)  
Publisher: IEEE

Show Title History

2020 International Conference on Mainstreaming Block Chain Implementation (ICOMBI)  
Publisher: IEEE

Show Title History

2020 International Conference on Omni-layer Intelligent Systems (COINS)  
Publisher: IEEE

Show Title History

2020 Signal Processing Workshop (SPW)  
Publisher: IEEE

Show Title History



**THE IEEE APP:**

Let's stay connected...

Download Today!

Available on the App Store | Get it on Google Play

IEEE



IEEE Authors:

Increase Your Research Impact

Add executable code to your research articles

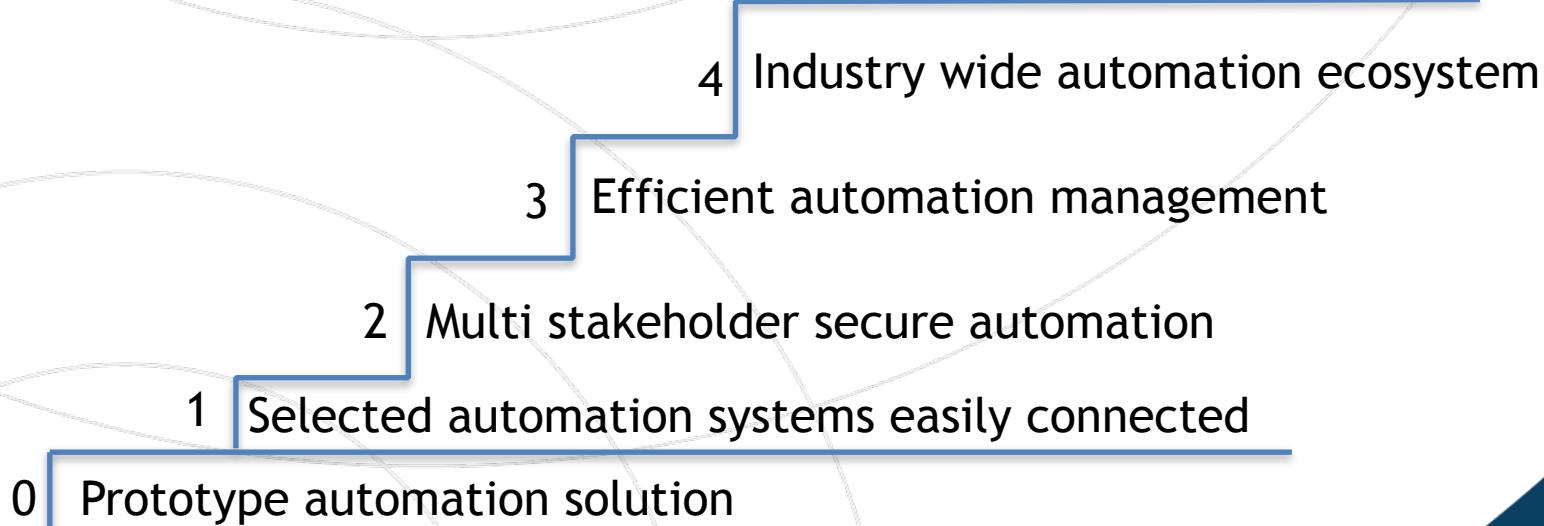
UPLOAD YOUR CODE



TOOLS

Automation engineer view

# Functionality stair



# 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

# Comments! Questions?

[jerker.delsing@ltu.se](mailto:jerker.delsing@ltu.se)