ENGINEERING OF COMPLEX SYSTEM OF SYSTEMS

Prof. Jerker Delsing
Luleå University of Technology
Sweden

Prof. Jerker Delsing

- Prof. Delsing and the EISLAB group http://www.ltu.se/eislab has been a partner of major EU projects in the field, e.g.
  - Socrades,
  - IMC-AESOP,
  - Arrowhead (coordinator),
  - FAR-EDGE (WP lead),
  - Productive4.0 (WP lead) and
  - Arrowhead Tools (coordinator).
- Delsing holds positions as vice president and board member of INSIDE (formerly ARTEMIS-IA) and board member of ProcessIT.EU and ProcessIT Innovations.

LULEÅ UNIVERSITY OF TECHNOLOGY

Engineering of complex System of Systems

Prof. Jerker Delsing

Luleå University of Technology

Sweden

This research work has been funded by the European Commission, through the European H2020 research and innovation programme, ECSEL Joint Undertaking, and National Funding Authorities from 18 involved countries under the research project Arrowhead Tools with Grant Agreement no. 826452.

ARROWHEAD
TOOLS

Complex System of Systems - SoS

Complex Cyber Physical System of Systems

Automation

Digitalisation

Industrie 4.0
Smart Home
INDUSTRY
CONSUMER
INFRASTRUCTURE
MOBILITY
ENERGY
Smart Car
Smart Grid

ARROWHEAD TOOLS

Plethora of standards to support engineering of automation solutions

IEC 62264, based on ANSI/ISA-95.

Competing standards in similar areas e.g.
IEC 61850, IEC 61970 and IEC 61968, primarily associated with power systems management.
ISO TC 184, collaborating with "Machinery Information Management Open Systems Alliance" (MIMOSA)
ISO 15926 - Industrial automation systems and integration, and
ISO 18435 - Industrial automation systems and integration
RAMI4.0

For life-cycle and hierarchical structure:

- IEC 62890 "Life-cycle management for systems and products used in industrial- process measurement, control and automation"
- IEC 62264 (ISA-95) / IEC 61512 (ISA-88)

For end-to-end engineering:

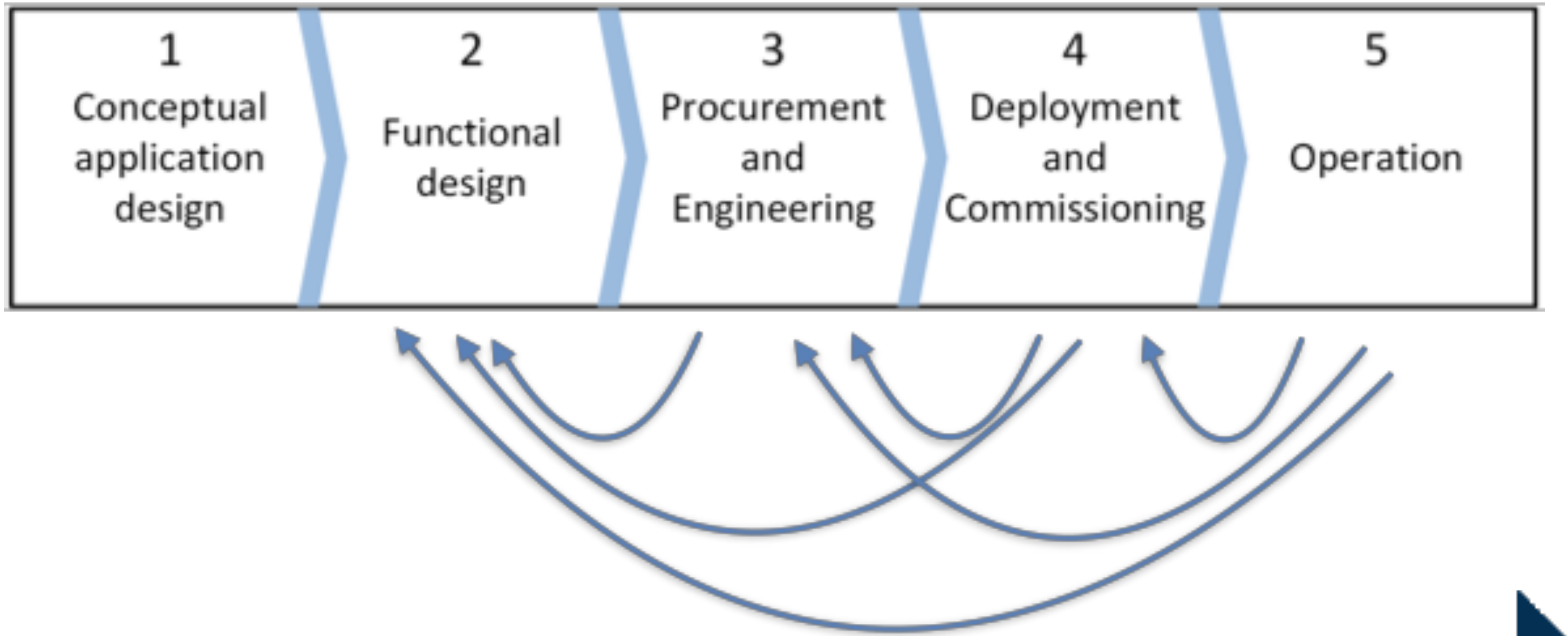
- AutomationML
- ProSTEP iViP
- eCl@ss

IEC 81346 "Industrial systems structuring principles"

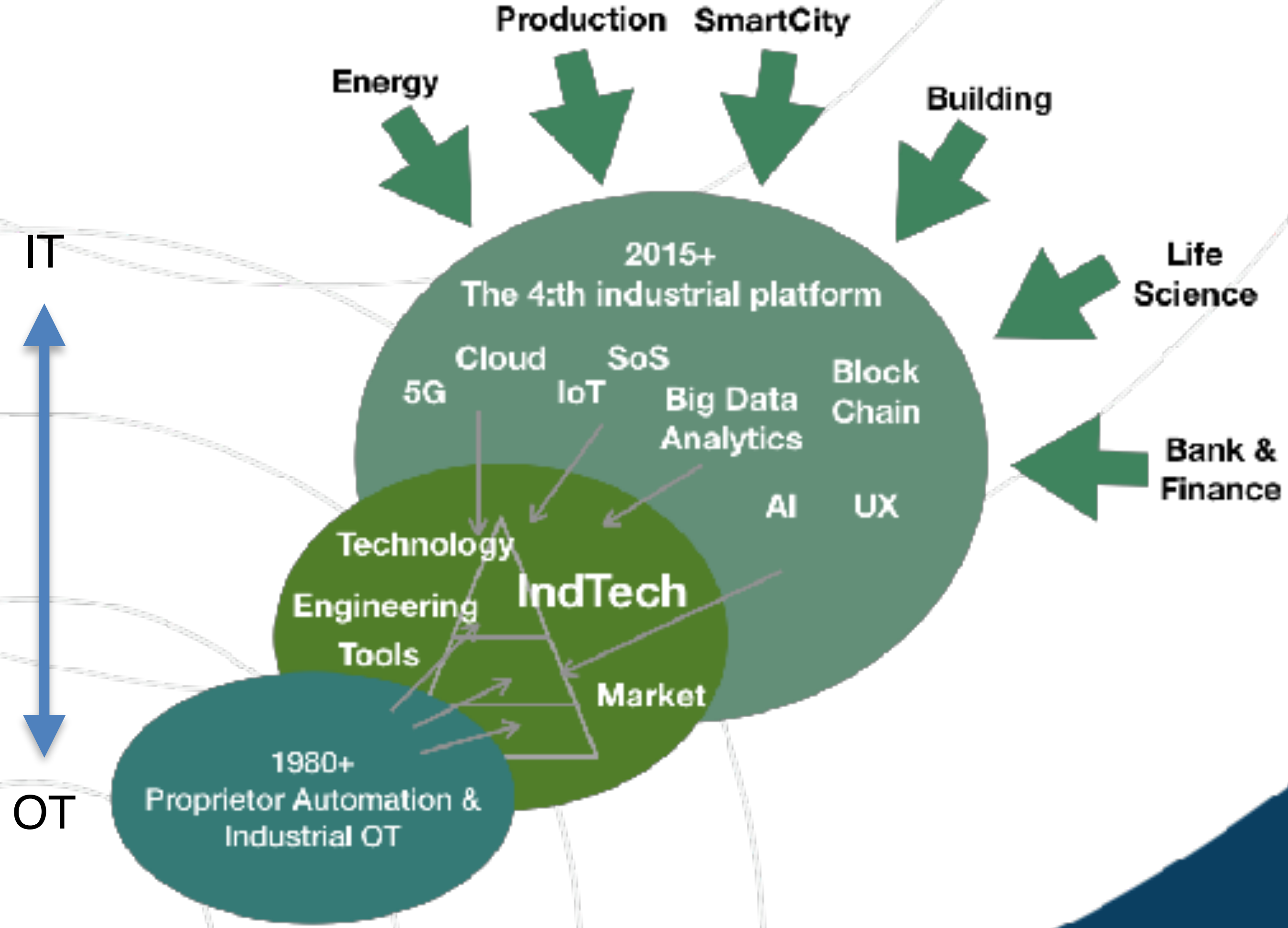
- IEC 62714, AutomationML
- IEC 62541 OPC-UA
- IEC 61131, IEC 61499, PLC coding

Basic engineering state of the art

IEC 81346

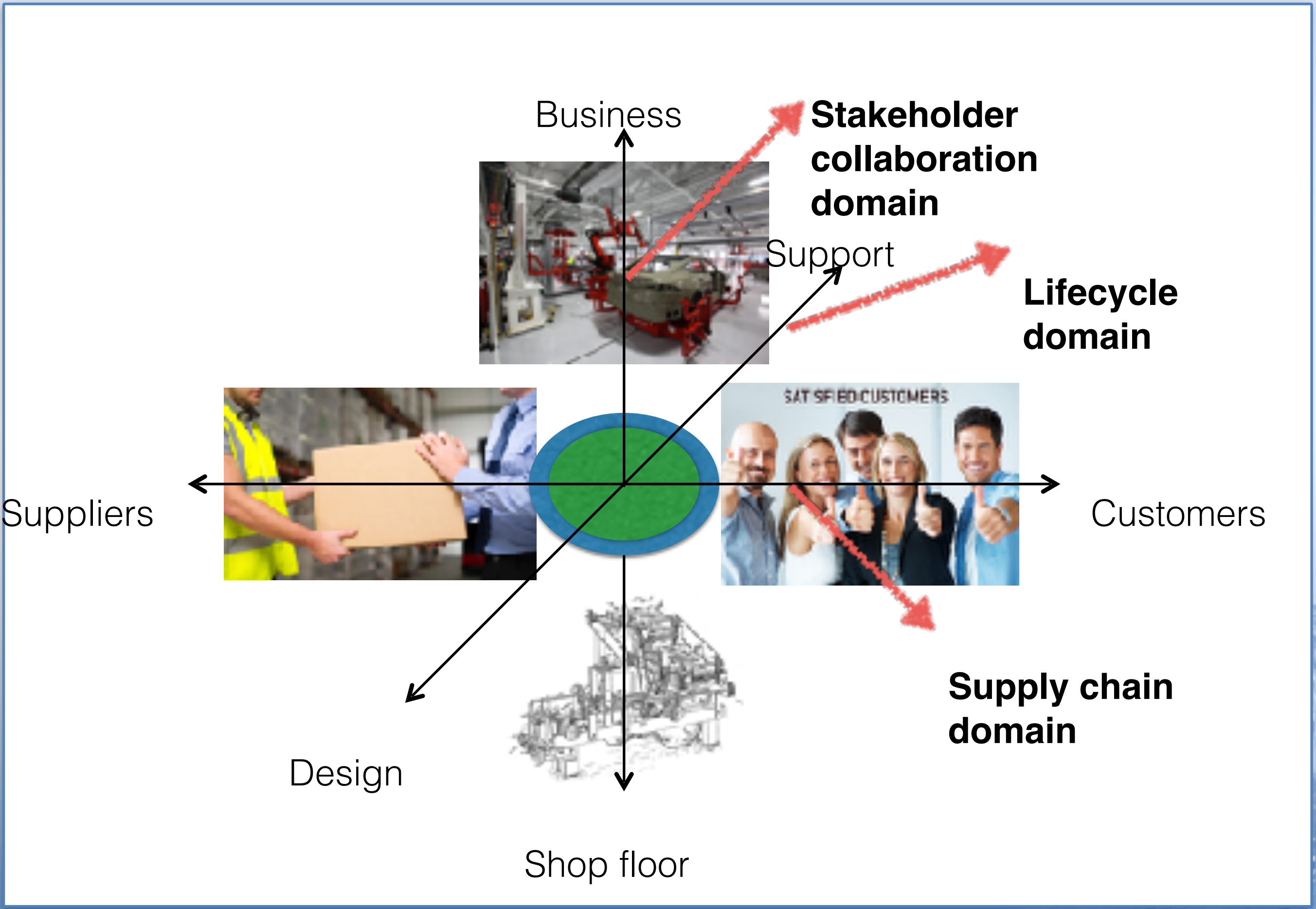

1
Conceptual application design
2
Functional design
3
Procurement and Engineering
4
Deployment and Commissioning
5
Operation

OT meets IT



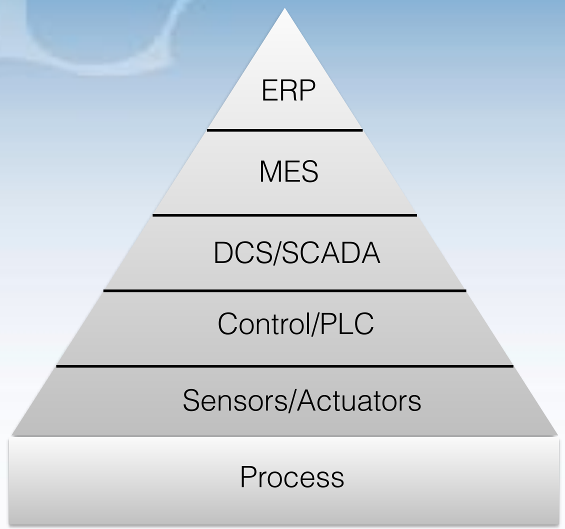
ARROWHEAD TOOLS

From enterprise to multi stakeholder operation

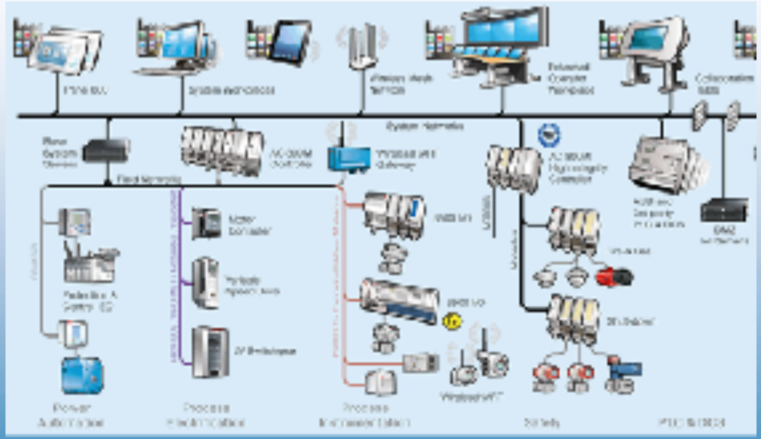


Current production automation

ISA-95



Hierarchical system implementation

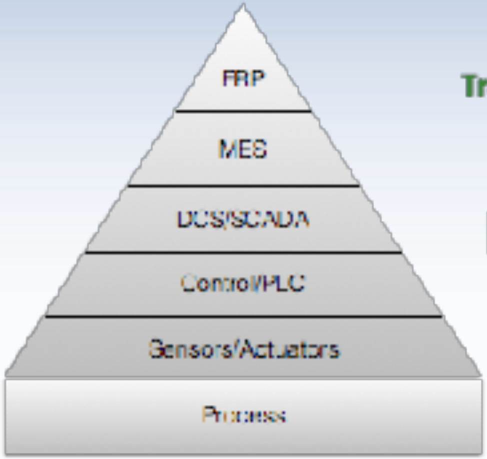


- Ridged pyramid
  - Inflexible automation
  - Cross layer dependencies
  - Low/No security

- Heterogeneous and incompatible networks
  - Industrial Ethernet
  - Fieldbus
  - Modbus
  - ASI bus
  - Hart/WirelessHart
  - 4-20 mA
  - ……

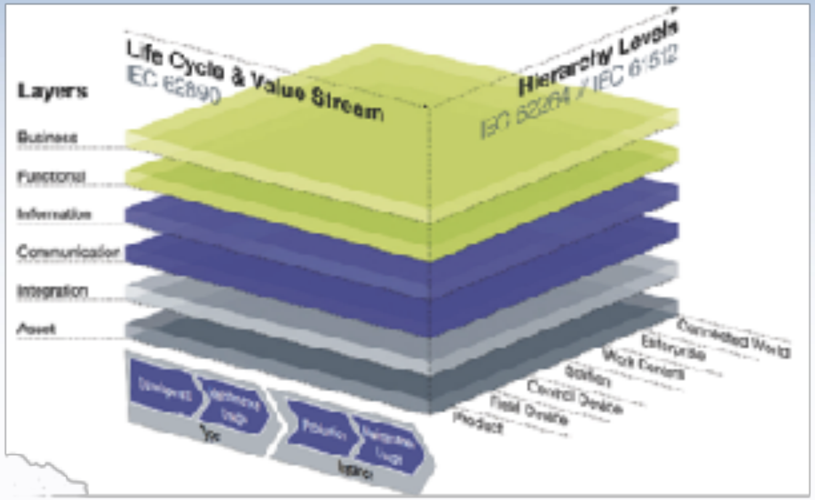
The automation technology transition

ISA 95



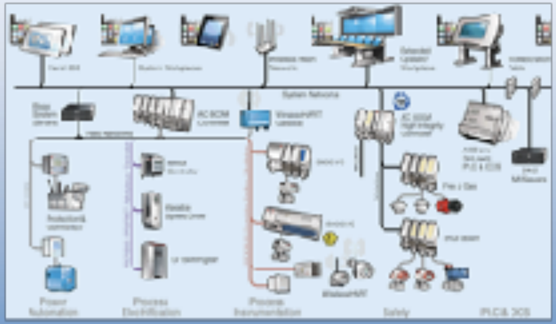
Transition path

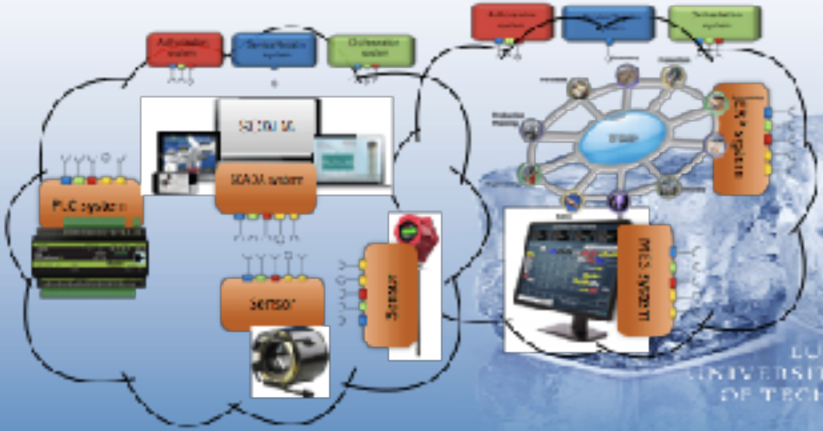
RAMI4.0



Hierarchical system implementation

Local automation cloud implementation



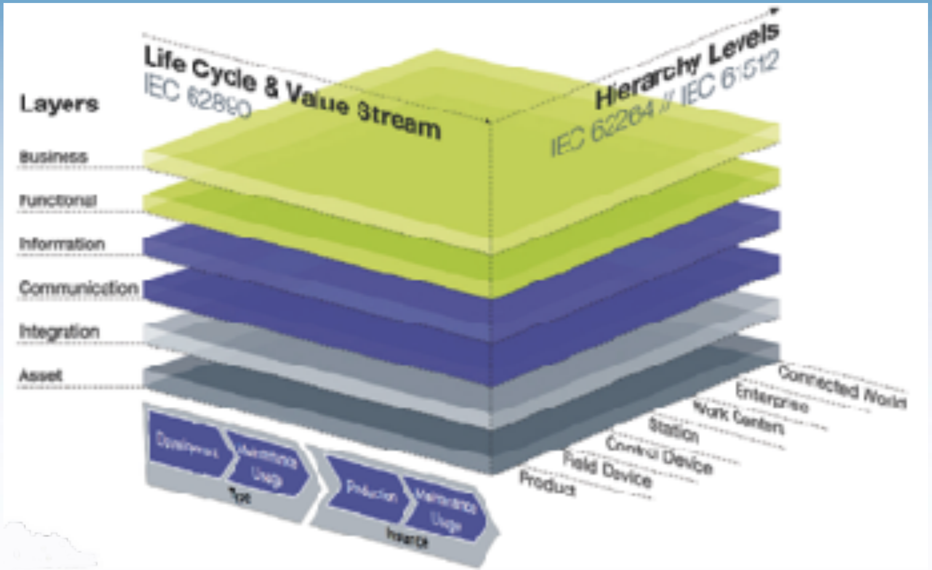


Digitised industry

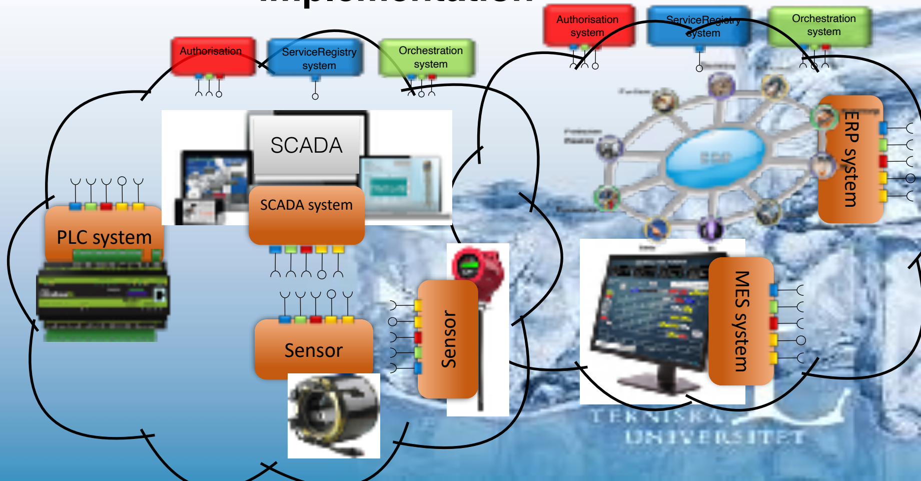
- Dynamic digital industry
  - Changes in run-time
  - High security

- System of Systems - IoT based
  - Interoperable IoT's
  - Functionality management
  - Security management

RAMI4.0



Local automation cloud implementation



Scalability

- Digitalisation is pushing for integration of more systems than today
  - Moving beyond 10^5 connected IoT’s

- Integration of today isolated systems
  - Preserving
    - Functionality
    - Real time
    - Security
    - Interoperability
    - …..

System of Systems integration to Cyber Physical System of Systems

- Service level integration
  - Descriptions of a plant
    - Physical functions
      - PI&D, ….
      - Control, ….
    - Electrical
      - Topology, logical
    - Communication, computation
      - Topology, Logical
    - Wiring
    - Layout

Digitalisation and Automation requirements

- Real time performance
- Engineering simplicity

- Interoperability
- Security and trust
- Safety
- Scalability
- System of Systems integration
- Flexibility

Scalability

- Digitalisation is pushing for integration of more systems than today
  - Moving beyond 10^5 connected IoT's
- Integration of today isolated systems
  - Preserving
    - Functionality
    - Real time
    - Security
    - Interoperability
  - Enabling
    - Maintenance
    - Evolution
    - Lifecycle management

Model based engineering - MBE

Modelling complex Cyber Physical System of Systems

Languages

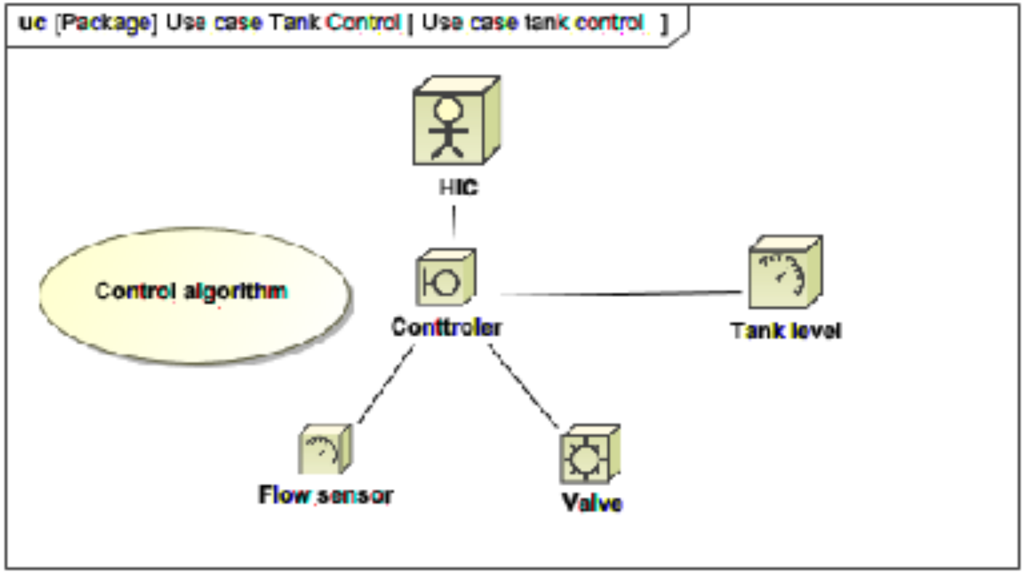
- UML - Cyber space
- SysML - Integrating cyber space and physics - CPS
- AutomationML - Control
- …

What is SysML

The **Systems Modeling Language** (**SysML**) 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.

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.



uc [Package] Use case Tank Control | Use case tank control]
HIC
Control algorithm
Conttroler
Tank level
Flow sensor
Valve

SysML Tools

MagicDraw - Cameo, commercial

SysML v1.6

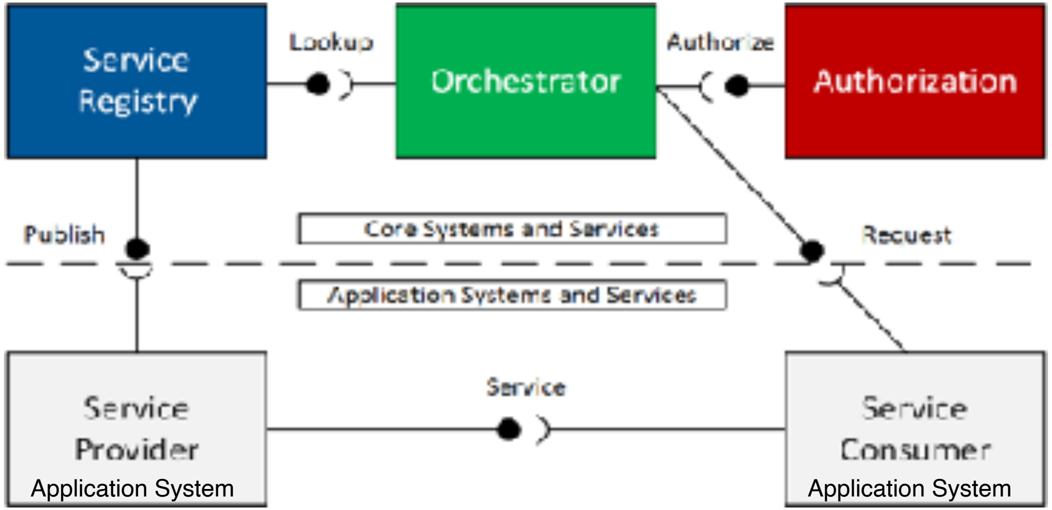
Extensive graphical system modelling tool based on SysML

Papyrus - Eclipse Open source

UML + SysML 1.6

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.

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

Key Arrowhead concepts to be modelled

- Network connecting
- Devices hosting
- SW-Systems constituting self contained
- Local clouds integrated to
- System of local clouds

SysML modelling basics

Requirement diagram/table

Use case diagrams

Activity diagram

Block definition diagrams

Internal block diagrams

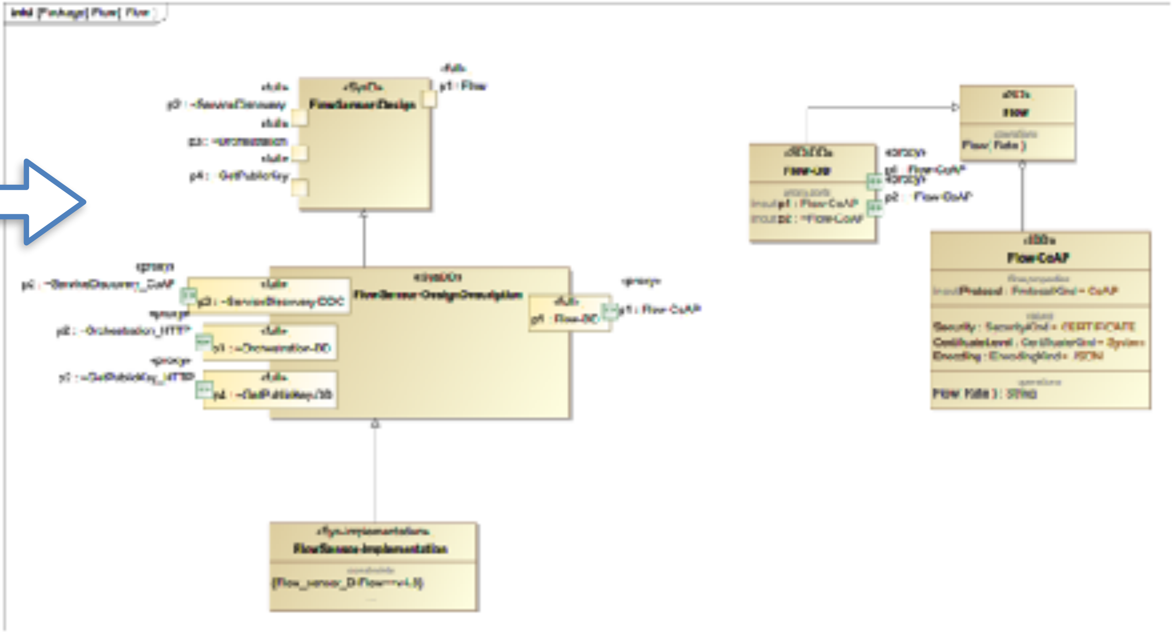
Parametric diagram

State machine diagram

Sequence diagrams

…

…



ARROWHEAD TOOLS

SOA SysML support

Library

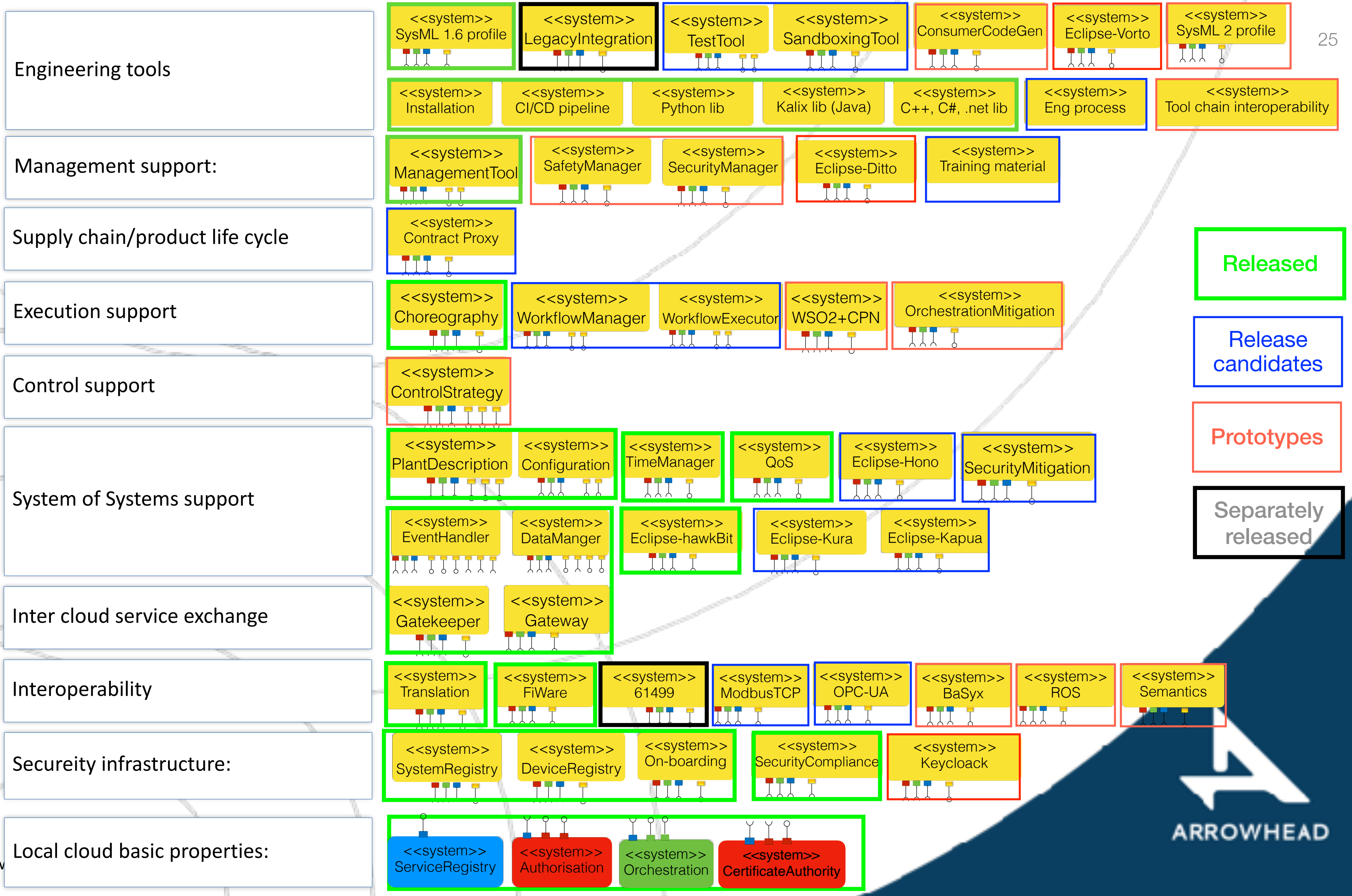
Eclipse Arrowhead core systems

Templates for

- Local clouds
- System of Local clouds
- Generic application systems
- Devices
- Network

www.github.com/eclipse-arrowhead

Eclipse Arrowhead v4.4.0



Engineering tools	<<system>> SysML 1.6 profile <<system>> LegacyIntegration <<system>> TestTool <<system>> SandboxingTool <<system>> ConsumerCodeGen <<system>> Eclipse-Vorto <<system>> SysML 2 profile <<system>> Installation <<system>> CI/CD pipeline <<system>> Python lib <<system>> Kalix lib (Java) <<system>> C++, C#, .net lib <<system>> Eng process <<system>> Tool chain interoperability
Management support:	<<system>> ManagementTool <<system>> SafetyManager <<system>> SecurityManager <<system>> Eclipse-Ditto <<system>> Training material
Supply chain/product life cycle	<<system>> Contract Proxy
Execution support	<<system>> Choreography <<system>> WorkflowManager <<system>> WorkflowExecutor <<system>> WSO2+CPN <<system>> OrchestrationMitigation
Control support	<<system>> ControlStrategy
System of Systems support	<<system>> PlantDescription <<system>> Configuration <<system>> TimeManager <<system>> QoS <<system>> Eclipse-Hono <<system>> SecurityMitigation <<system>> EventHandler <<system>> DataManger <<system>> Eclipse-hawkBit <<system>> Eclipse-Kura <<system>> Eclipse-Kapua
Inter cloud service exchange	<<system>> Gatekeeper <<system>> Gateway
Interoperability	<<system>> Translation <<system>> FiWare <<system>> 61499 <<system>> ModbusTCP <<system>> OPC-UA <<system>> BaSyx <<system>> ROS <<system>> Semantics
Secureity infrastructure:	<<system>> SystemRegistry <<system>> DeviceRegistry <<system>> On-boarding <<system>> SecurityCompliance <<system>> Keycloack
Local cloud basic properties:	<<system>> ServiceRegistry <<system>> Authorisation <<system>> Orchestration <<system>> CertificateAuthority

- 🟩 Released
- 🟦 Release candidates
- 🟥 Prototypes
- ⬛ Separately released

ARROWHEAD

ww

SOA support

SysML Profile

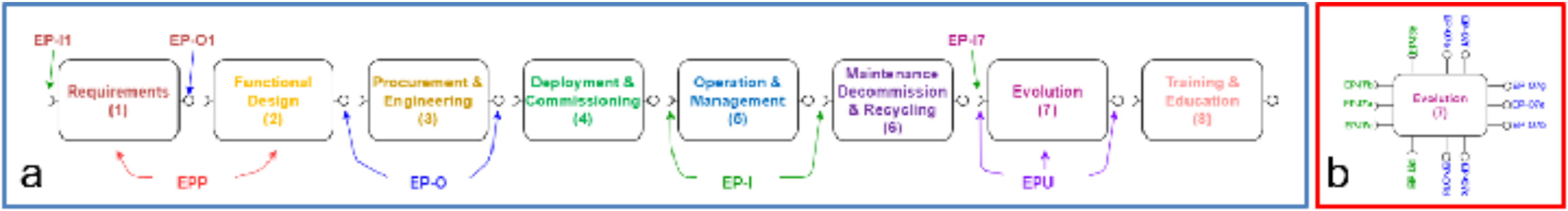
Based on Eclipse Arrowhead

Intend to support several engineering phases for a solution

- Requirement
- Design conceptual, black box,
- Design of implementation, white box,
- Procurement & Engineering
- Deployment
- Maintenance
- Evolution

Integration with the engineering process

Modelling the engineering process



EP-I1
EP-O1
EP-I7

Requirements (1)

Functional Design (2)

Procurement & Engineering (3)

Deployment & Commissioning (4)

Operation & Management (5)

Maintenance Decommission & Recycling (6)

Evolution (7)

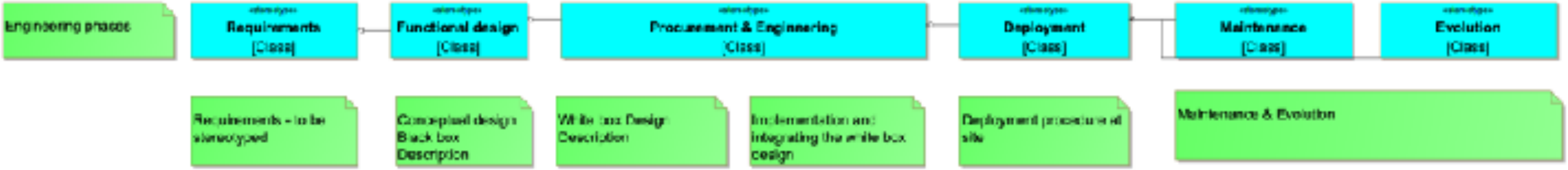
Training & Education (8)

EPP
EP-O
EP-I
EPU

a

Evolution (7)

b



Engineering phases

«stereotype» Requirements [Class]

«stereotype» Functional design [Class]

«stereotype» Procurement & Engineering [Class]

«stereotype» Deployment [Class]

«stereotype» Maintenance [Class]

«stereotype» Evolution [Class]

Requirements – to be stereotyped

Conceptual design Black box Description

White box Design Description

Implementation and integrating the white box design

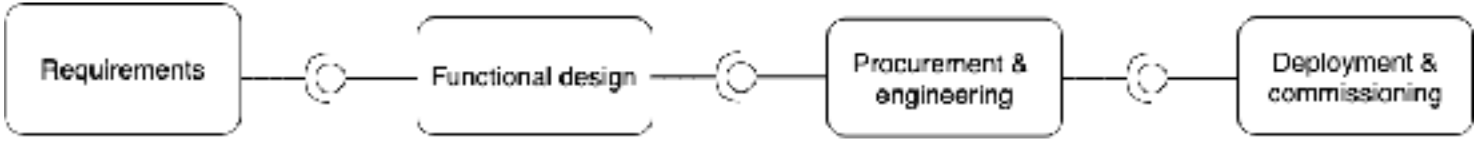
Deployment procedure at site

Maintenance & Evolution

ARROWHEAD TOOLS

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,

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,

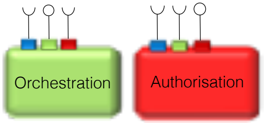
LCDD, SysDD, IDD

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

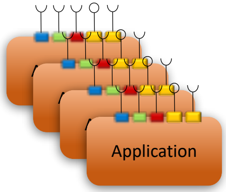
Commissioning of:
- Local cloud functionality
- System of local cloud functionality

ARROWHEAD FRAMEWORK LOCAL CLOUD

Core system selection



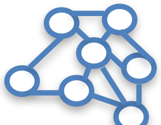
Application system & service design

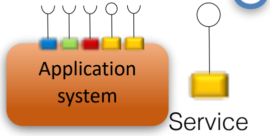


Devices and network

OS

Network



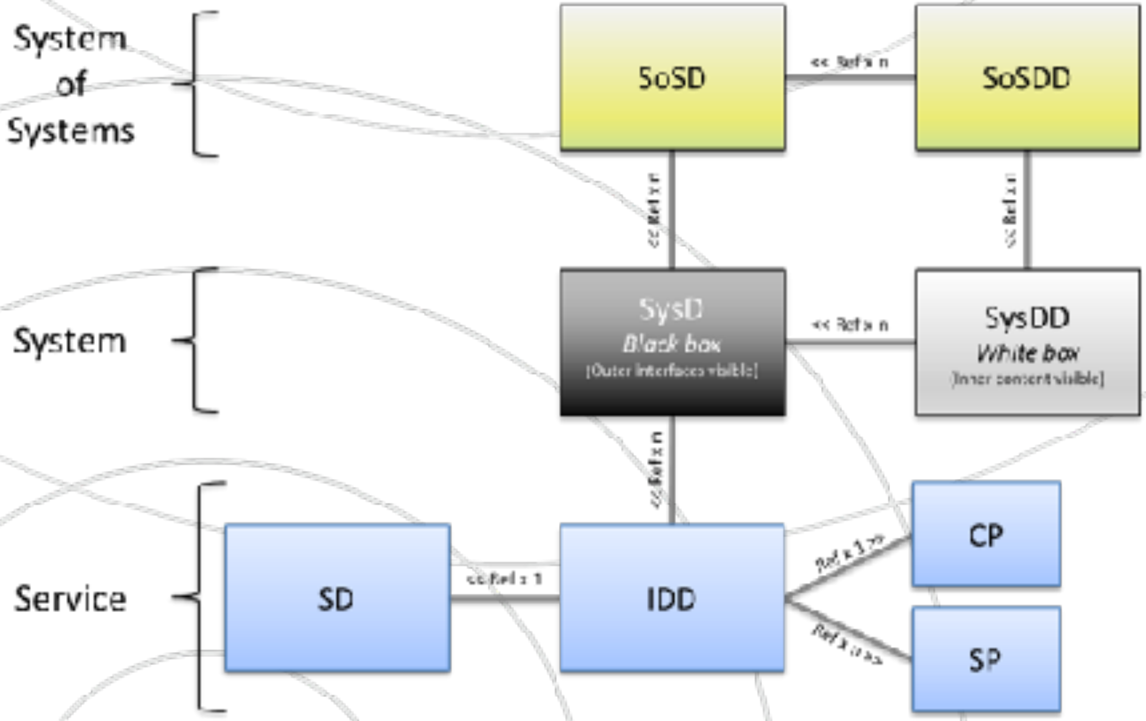


Physical deployment at site
- **Devices**
- **Routers**
- **Power supply**
- **Network connection**
-

www.arrowhead.eu

ARROWHEAD TOOLS

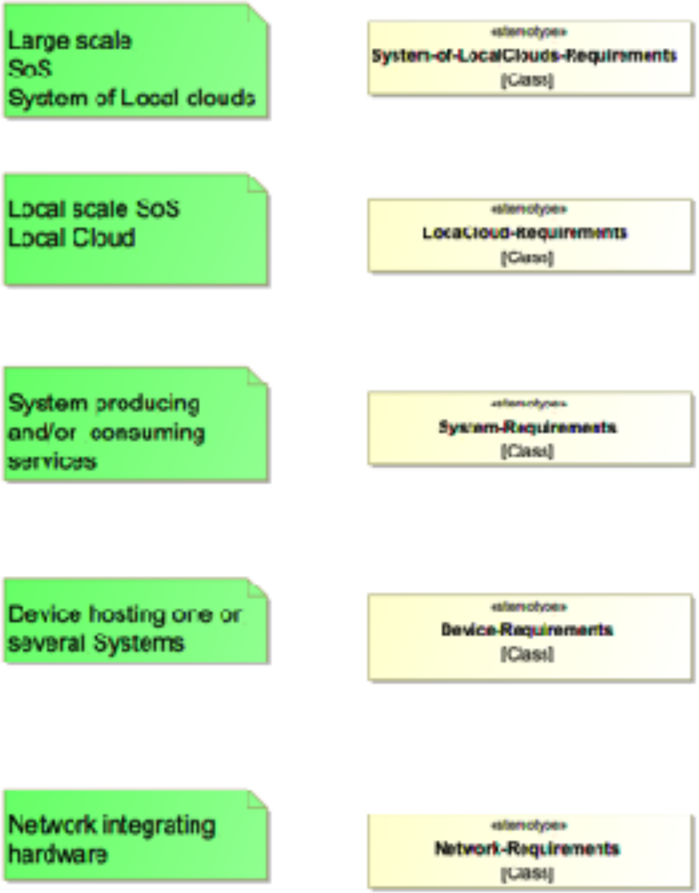
Engineering tools for cloud automation systems
Development support, documentation.



SoSD:	System-of-Systems Description
SoSDD:	System of Systems Design Description
SysD:	System Description
SysDD:	System Design Description
SD:	Service Description
IDD:	Interface Design Description
CP:	Communication Profile
SP:	Semantic Profile

ARROWHEAD

Architecture modelling



Large scale
SoS
System of Local clouds

«stereotype»
System-of-LocalClouds-Requirements
[Class]

Local scale SoS
Local Cloud

«stereotype»
LocalCloud-Requirements
[Class]

System producing
and/or consuming
services

«stereotype»
System-Requirements
[Class]

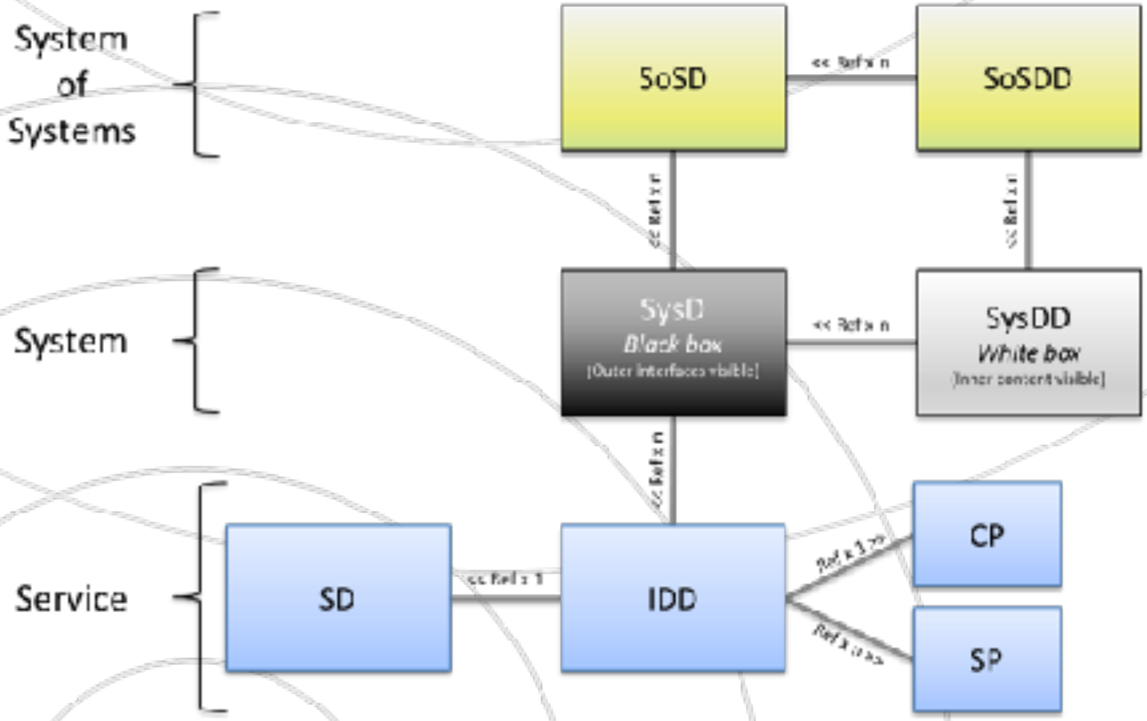
Device hosting one or
several Systems

«stereotype»
Device-Requirements
[Class]

Network integrating
hardware

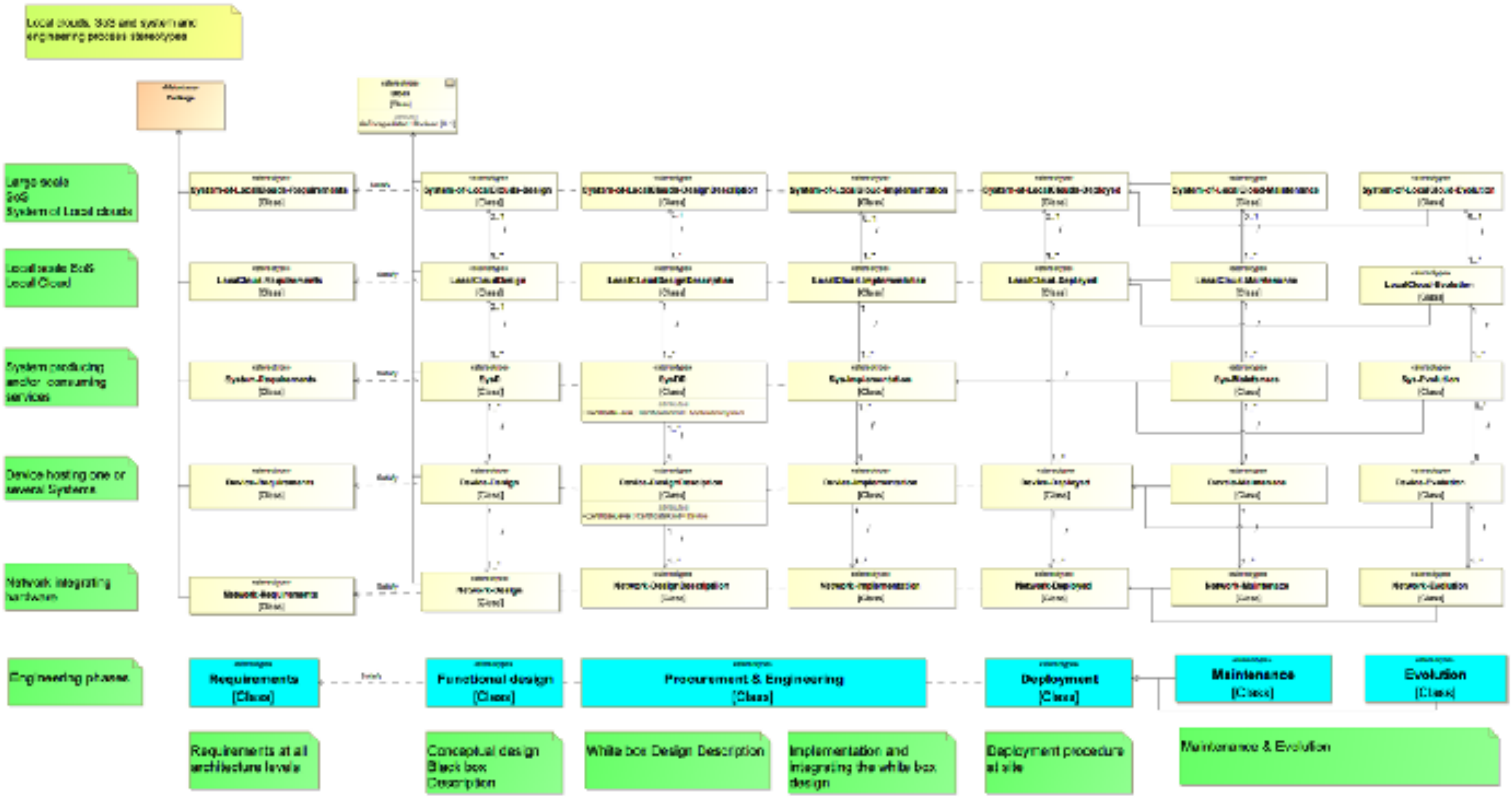
«stereotype»
Network-Requirements
[Class]

Eclipse Arrowhead documentation structure



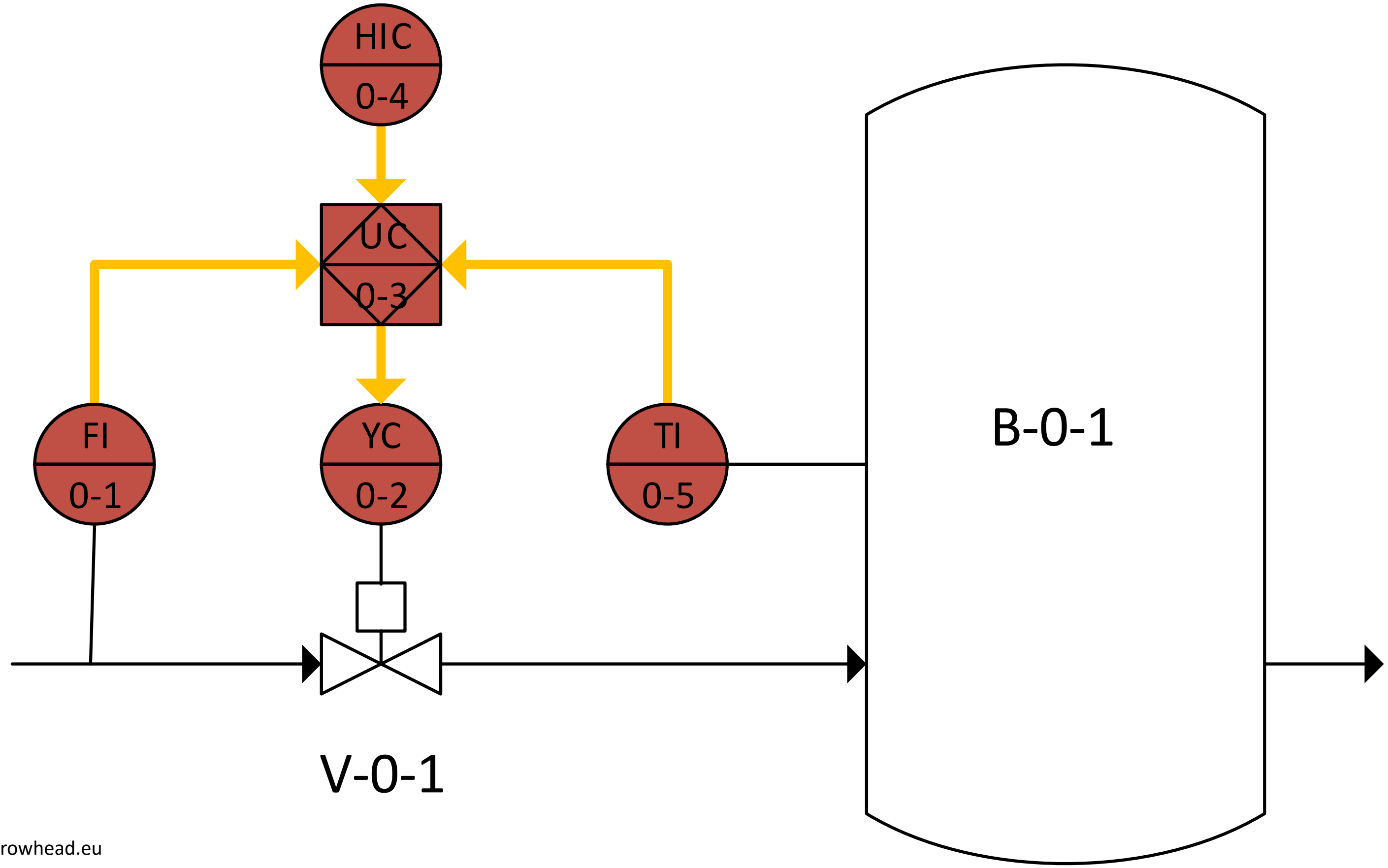
SoSD: System-of-Systems Description
SoSDD: System of Systems Design Description
SysD: System Description
SysDD: System Design Description
SD: Service Description
IDD: Interface Design Description
CP: Communication Profile
SP: Semantic Profile

SoS architecture and engineering in SysML

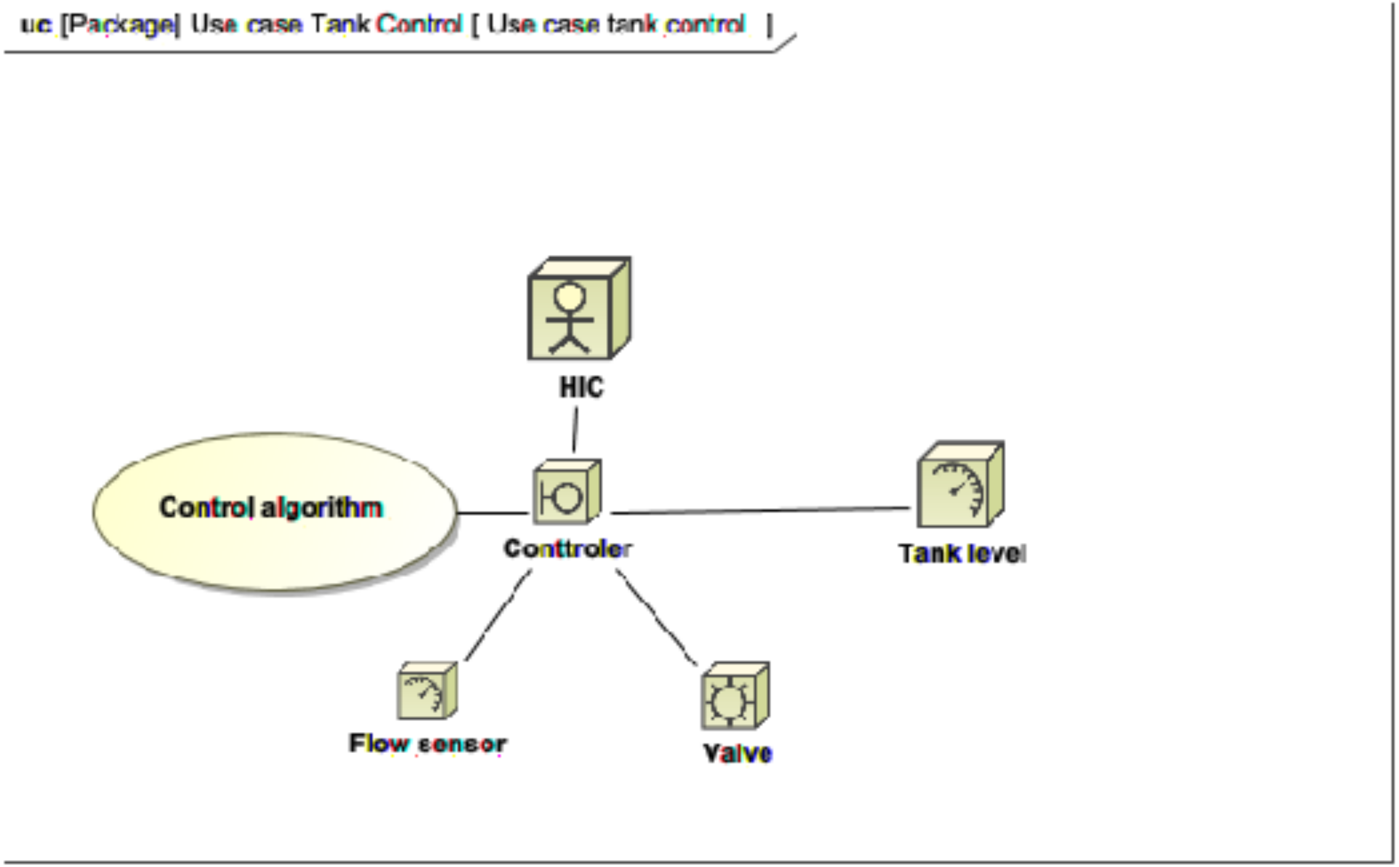

Local clouds, SoS and system and engineering process stereotypes
Large scale SoS System of Local clouds
System-of-LocalClouds-Requirements [Class]
System-of-LocalClouds-Design [Class]
System-of-LocalClouds-DesignDescription [Class]
System-of-LocalCloud-Implementation [Class]
System-of-LocalClouds-Deployed [Class]
System-of-LocalCloud-Maintenance [Class]
System-of-LocalCloud-Evolution [Class]
Local scale SoS Local Cloud
LocalCloud-Requirements [Class]
LocalCloudDesign [Class]
LocalCloudDesignDescription [Class]
LocalCloud-Implementation [Class]
LocalCloud-Deployed [Class]
LocalCloud-Maintenance [Class]
LocalCloud-Evolution [Class]
System producing and/or consuming services
System-Requirements [Class]
SysD [Class]
SysDD [Class]
SysImplementation [Class]
SysMaintenance [Class]
SysEvolution [Class]
Device hosting one or several Systems
Device-Requirements [Class]
Device-Design [Class]
Device-DesignDescription [Class]
Device-Implementation [Class]
Device-Deployed [Class]
Device-Maintenance [Class]
Device-Evolution [Class]
Network integrating hardware
Network-Requirements [Class]
Network-Design [Class]
Network-DesignDescription [Class]
Network-Implementation [Class]
Network-Deployed [Class]
Network-Maintenance [Class]
Network-Evolution [Class]
satisfy
Engineering phases
Requirements [Class]
Functional design [Class]
Procurement & Engineering [Class]
Deployment [Class]
Maintenance [Class]
Evolution [Class]
Requirements at all architecture levels
Conceptual design Black box Description
White box Design Description
Implementation and integrating the white box design
Deployment procedure at site
Maintenance & Evolution

TOOLS

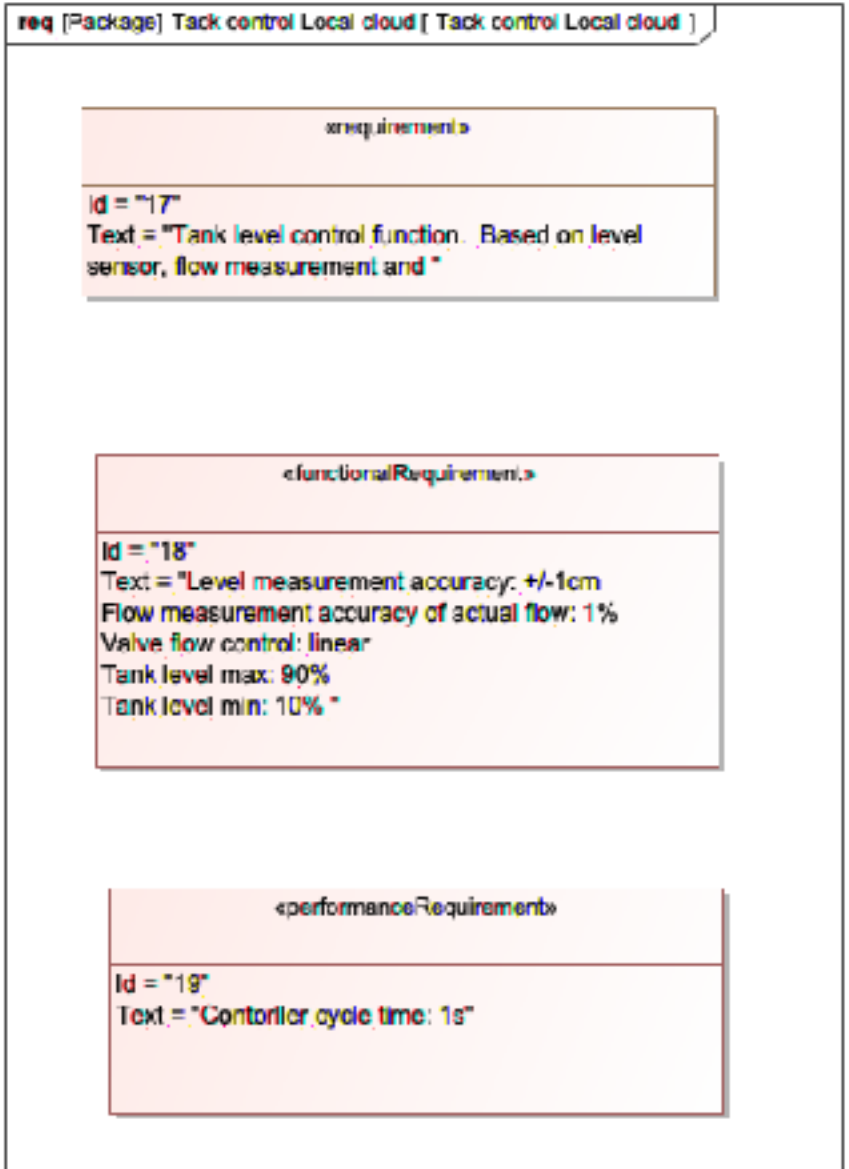
Lets start with a use case



Use case diagram


uc [Package] Use case Tank Control [Use case tank control]
HIC
Control algorithm
Conttroler
Tank level
Flow sensor
Valve

Requirements



req [Package] Tack control Local cloud [Tack control Local cloud]

«requirement»

Id = "17"
Text = "Tank level control function. Based on level sensor, flow measurement and "

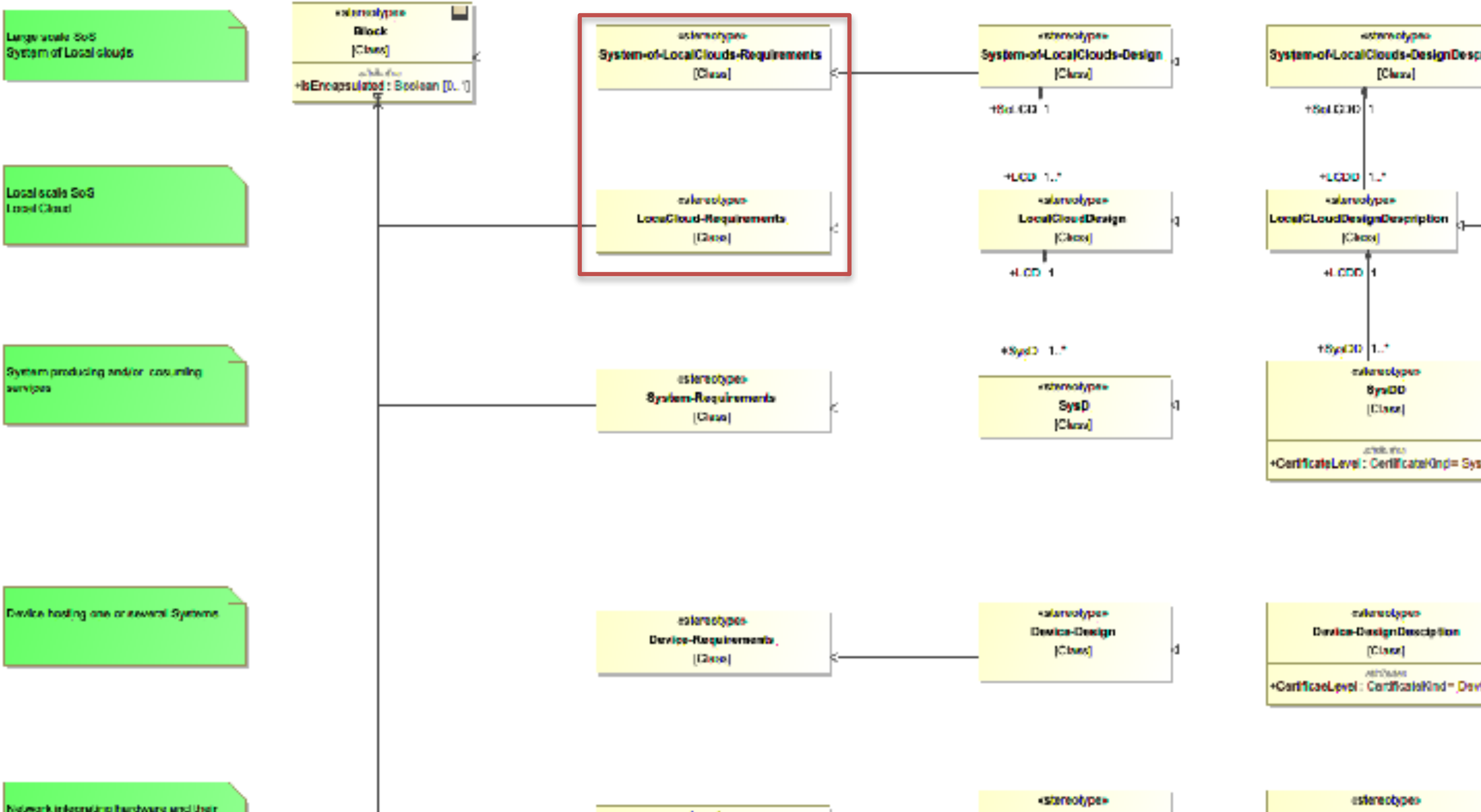
«functionalRequirement»

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 = "Contorller cycle time: 1s"

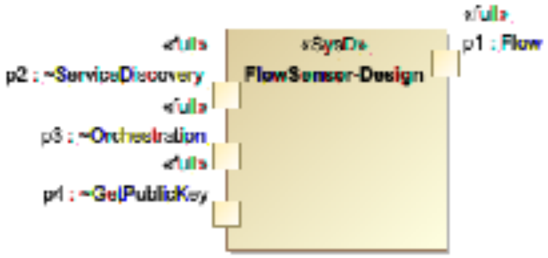
SoS architecture and engineering in SysML

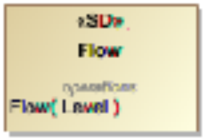


Functional system and service design

Micro-systems

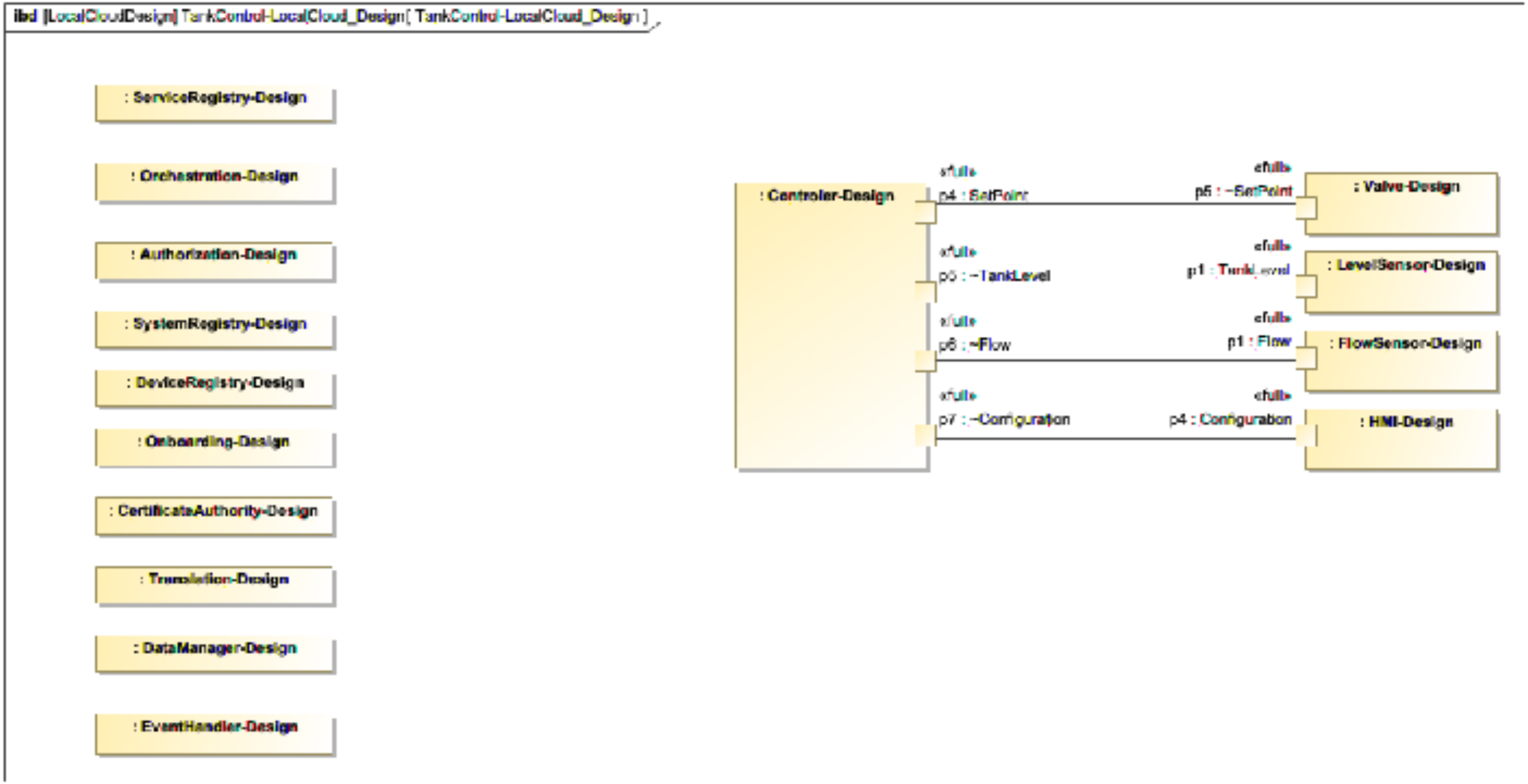
Micro-services



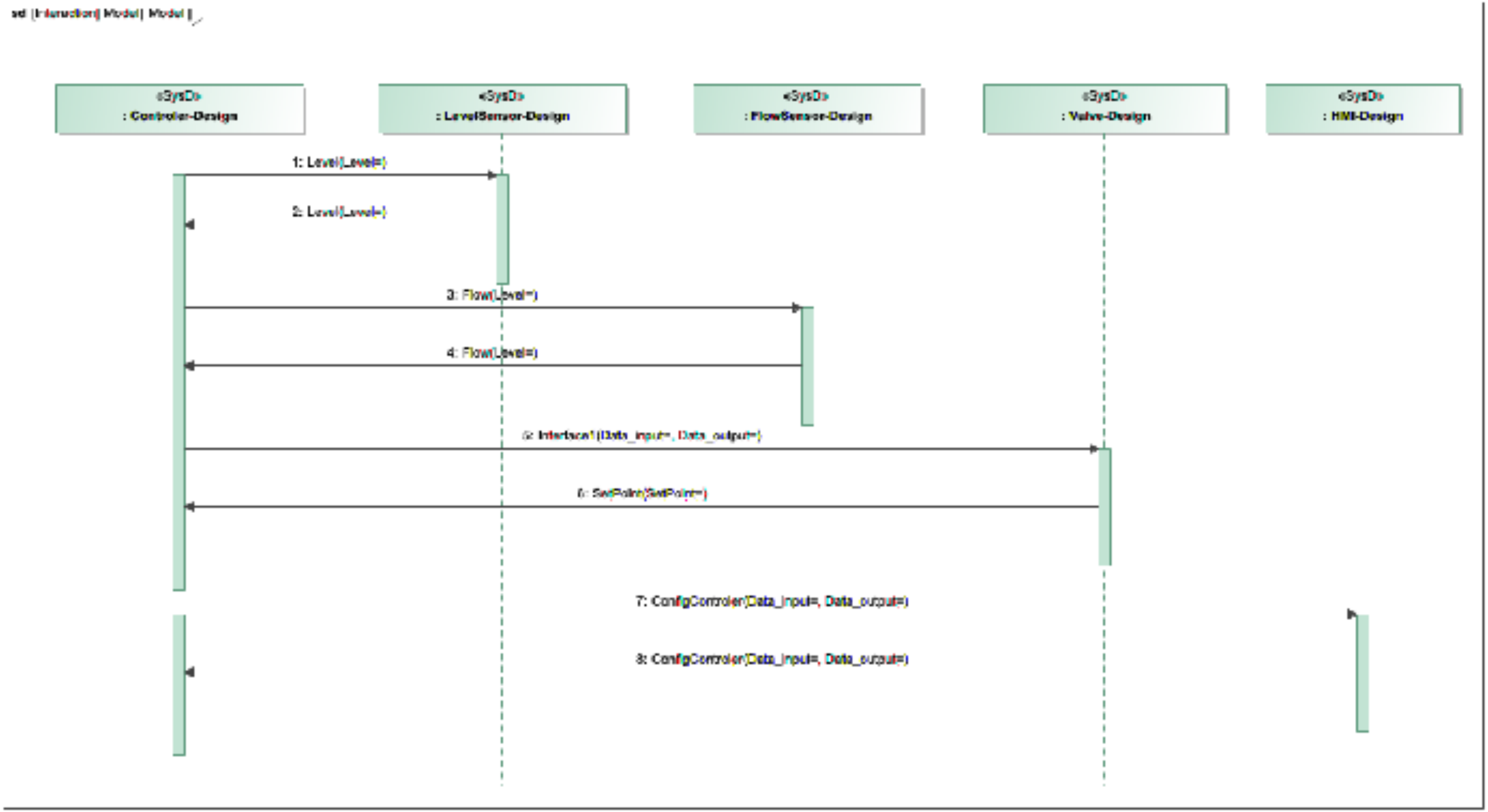


Functional SoS design - black box

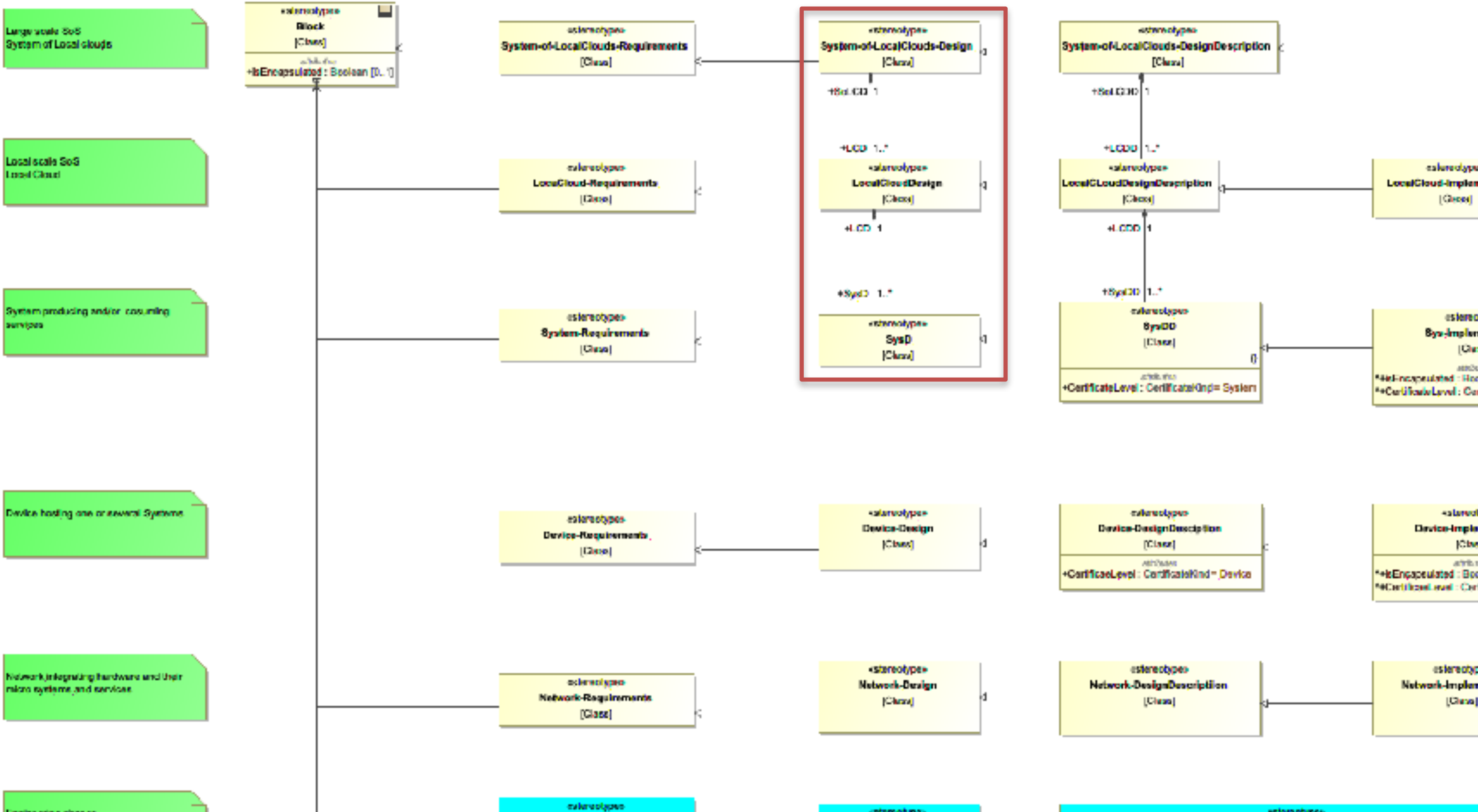
Local cloud functional orchestration



Service exchange functionality



SoS architecture and engineering in SysML



White box engineering

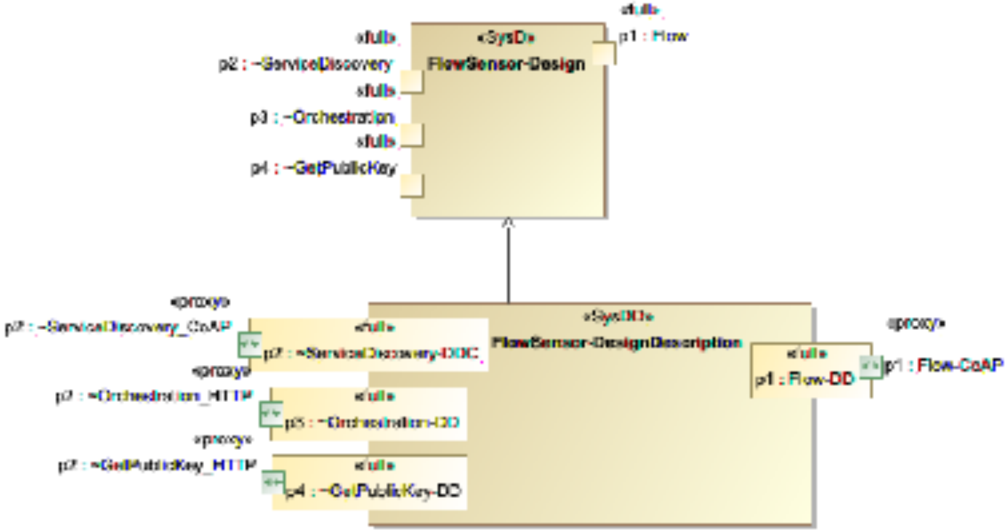
SysDD and IDD

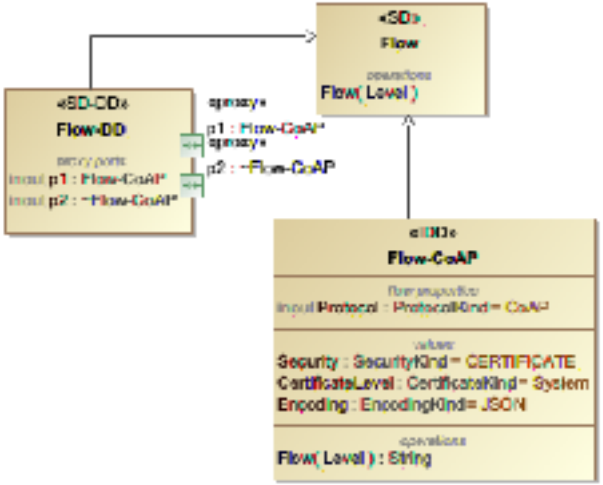
ARROWHEAD TOOLS

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

Micro-systems

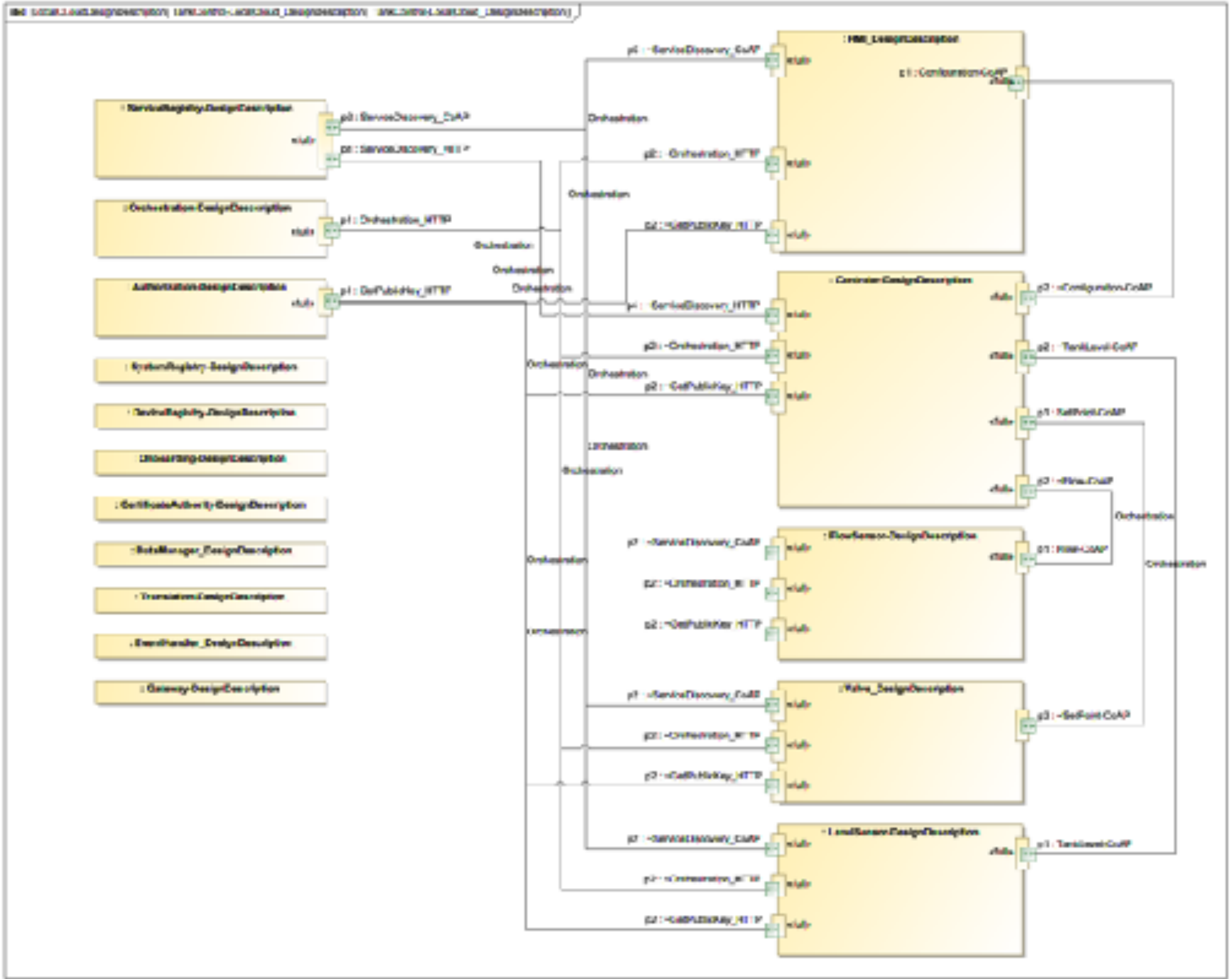
Micro-services





Functional LC design description - white box

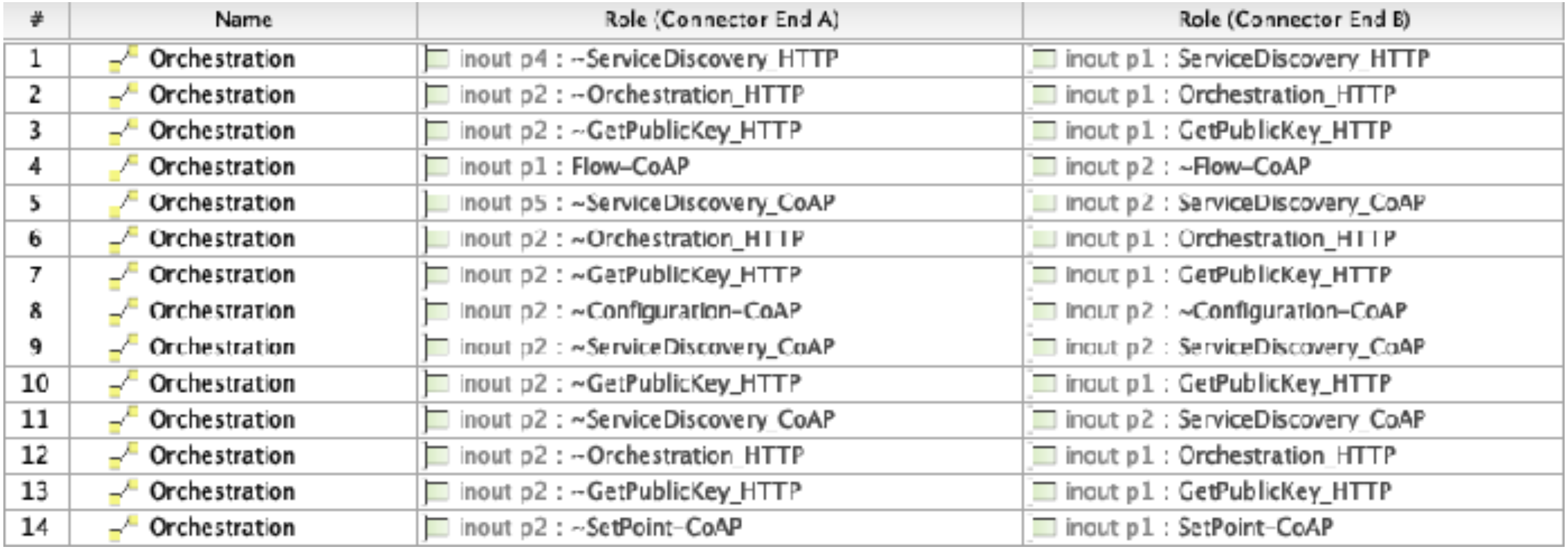
Local cloud functional design description model



: ServiceRegistry_DesignDescription
: Orchestration_DesignDescription
: Authorization_DesignDescription
: SystemRegistry_DesignDescription
: DeviceRegistry_DesignDescription
: Onboarding_DesignDescription
: CertificateAuthority_DesignDescription
: DataManager_DesignDescription
: Translation_DesignDescription
: EventHandler_DesignDescription
: Gateway_DesignDescription
: HMI_DesignDescription
: Controller_DesignDescription
: FlowSensor_DesignDescription
: Valve_DesignDescription
: LevelSensor_DesignDescription
p3 : ServiceDiscovery_CoAP
p2 : ServiceDiscovery_HTTP
p1 : Orchestration_HTTP
p1 : GetPublicKey_HTTP
p1 : ~ServiceDiscovery_CoAP
p2 : ~Orchestration_HTTP
p2 : ~GetPublicKey_HTTP
p1 : Configuration-CoAP
p3 : ~Configuration-CoAP
p1 : ~ServiceDiscovery_HTTP
p2 : TankLevel-CoAP
p1 : SetPoint-CoAP
p2 : ~Flow-CoAP
p1 : Flow-CoAP
p3 : ~SetPoint-CoAP
p1 : TankLevel-CoAP
Orchestration
«full»

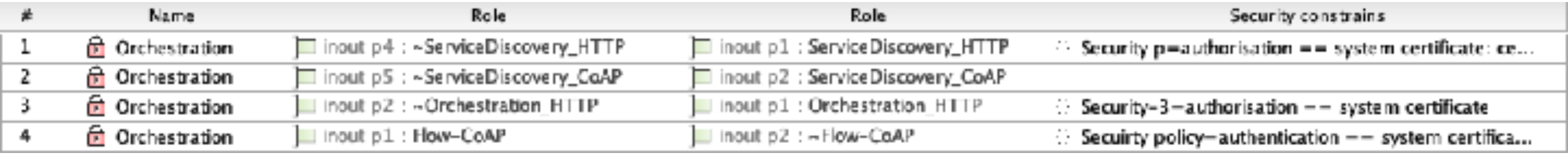
www.arrowhead.eu

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

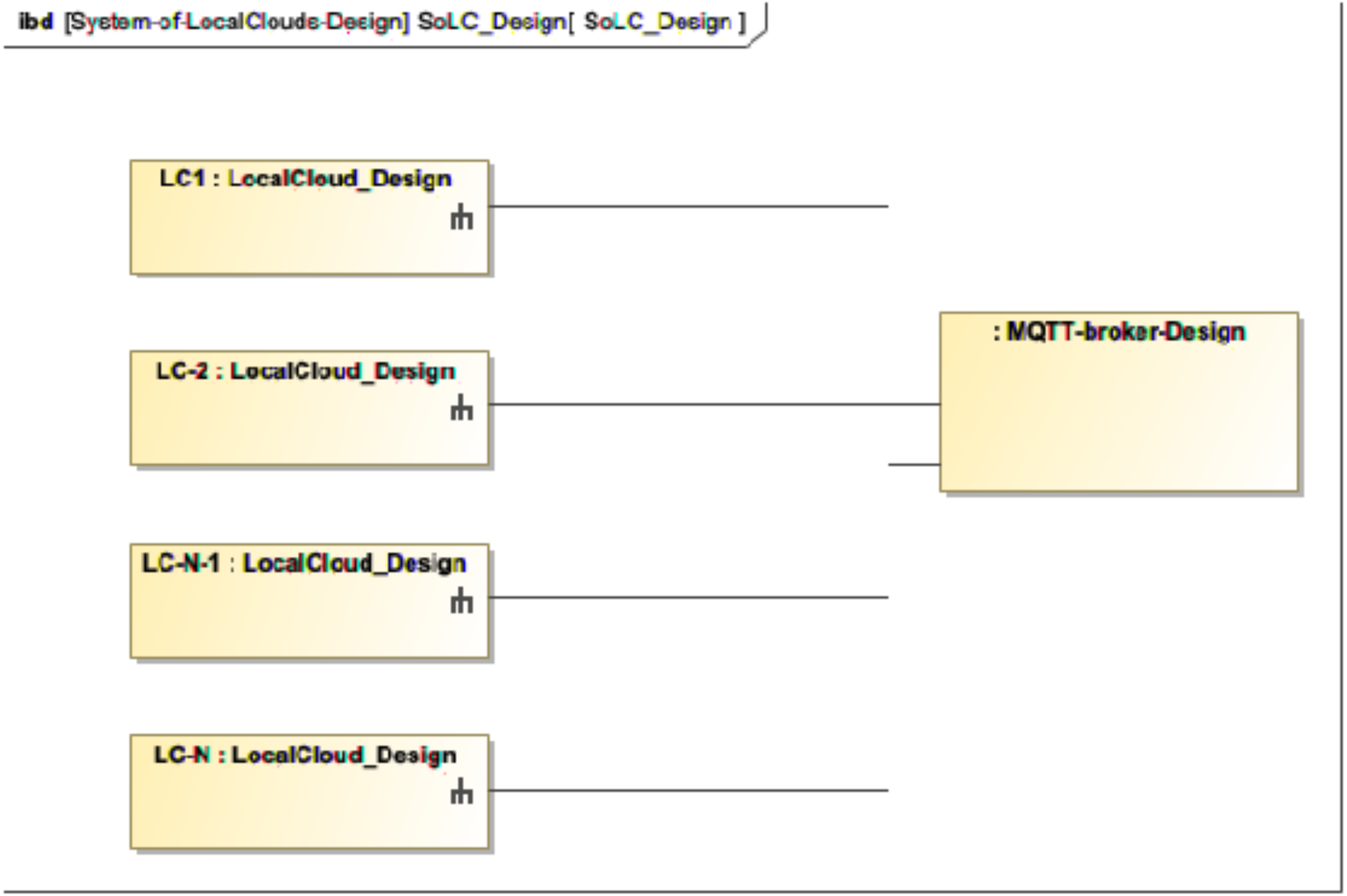
Security policys - rules and conditions



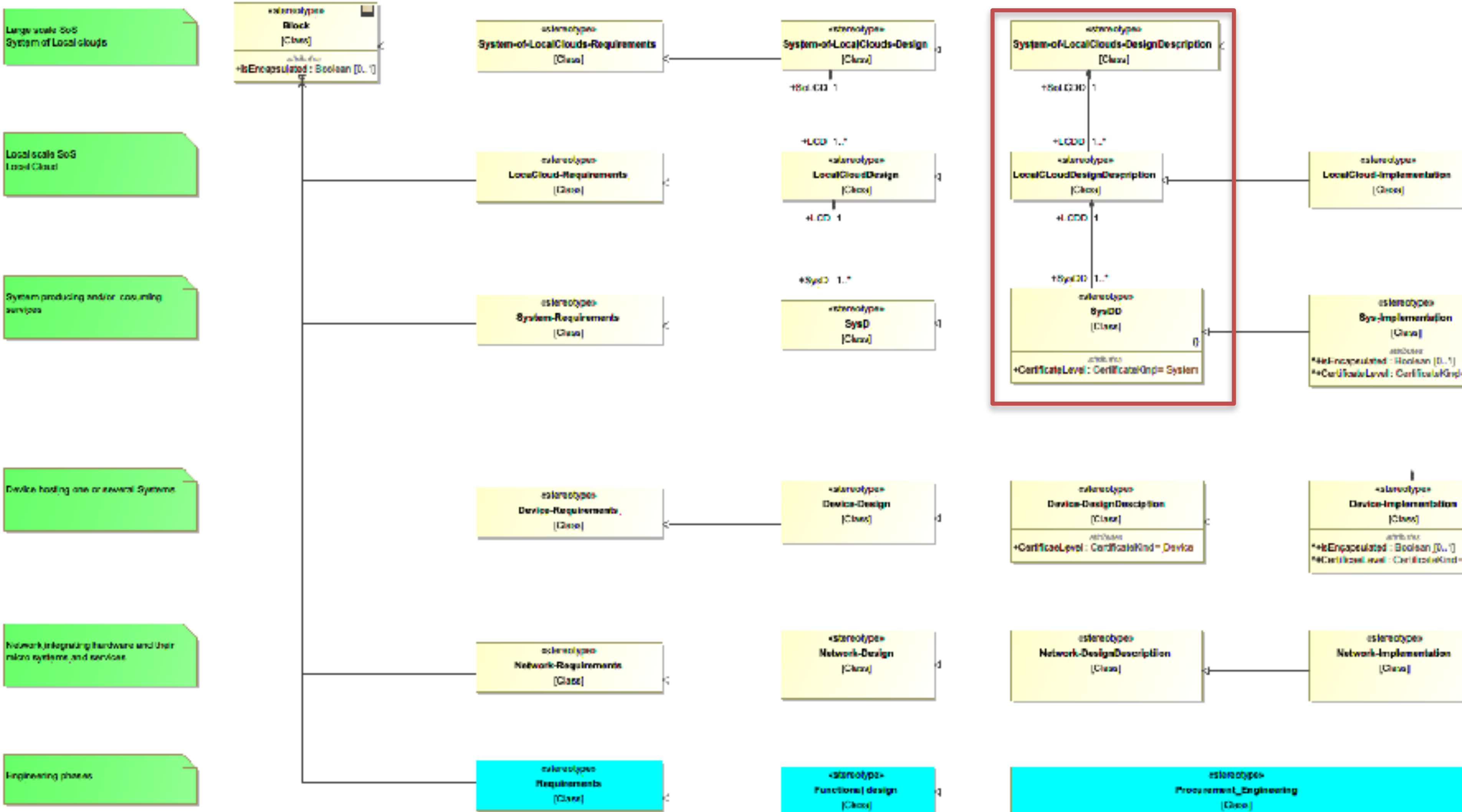
#	Name	Role	Role	Security constrains
1	Orchestration	inout p4 : ~ServiceDiscovery_HTTP	inout p1 : ServiceDiscovery_HTTP	Security p=authorisation == system certificate: ce...
2	Orchestration	inout p5 : ~ServiceDiscovery_CoAP	inout p2 : ServiceDiscovery_CoAP	
3	Orchestration	inout p2 : ~Orchestration_HTTP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

Functional SoLC design

System o local clouds functional design model



SoS architecture and engineering in SysML



Implementation

We also need

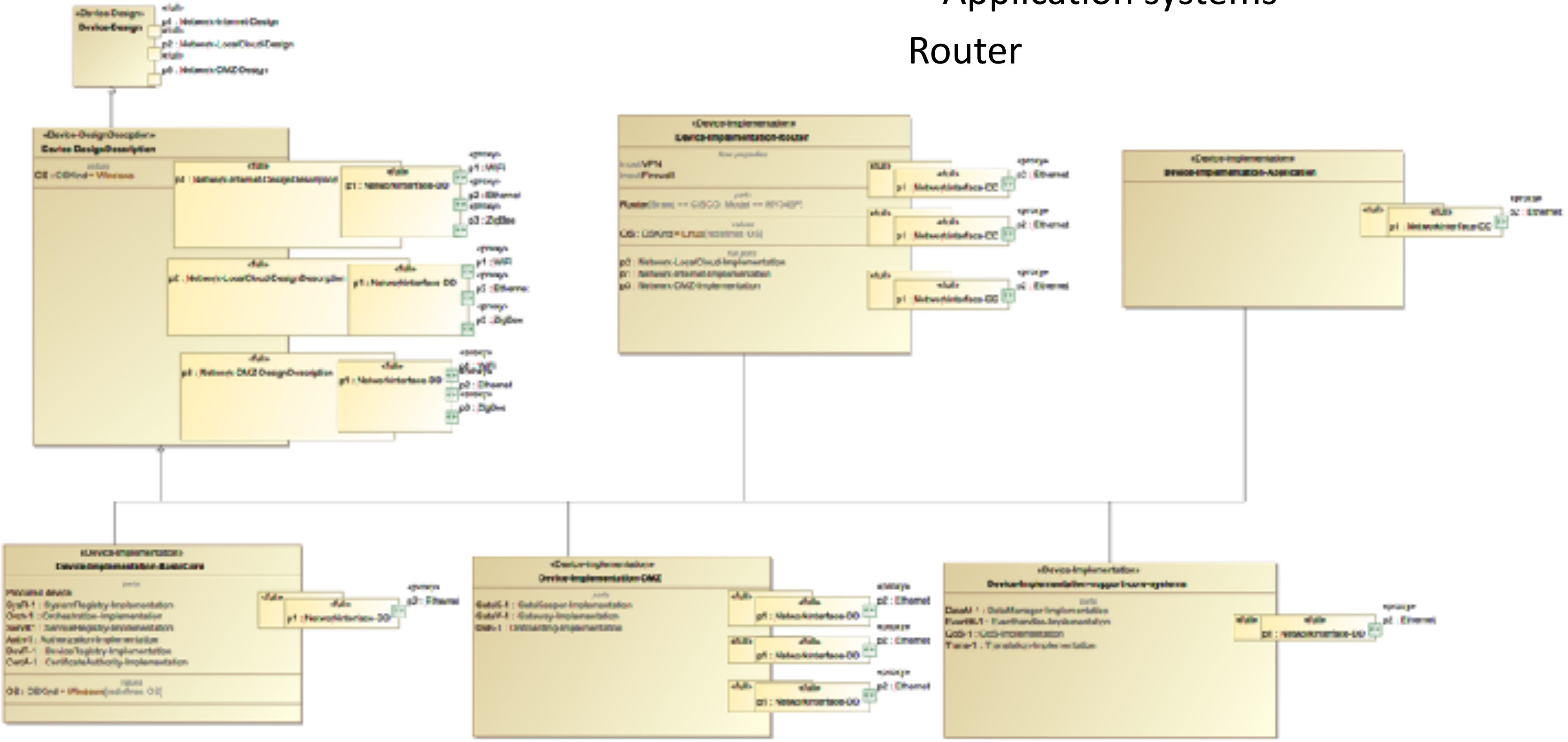
- Devices
- Network

Device implementation

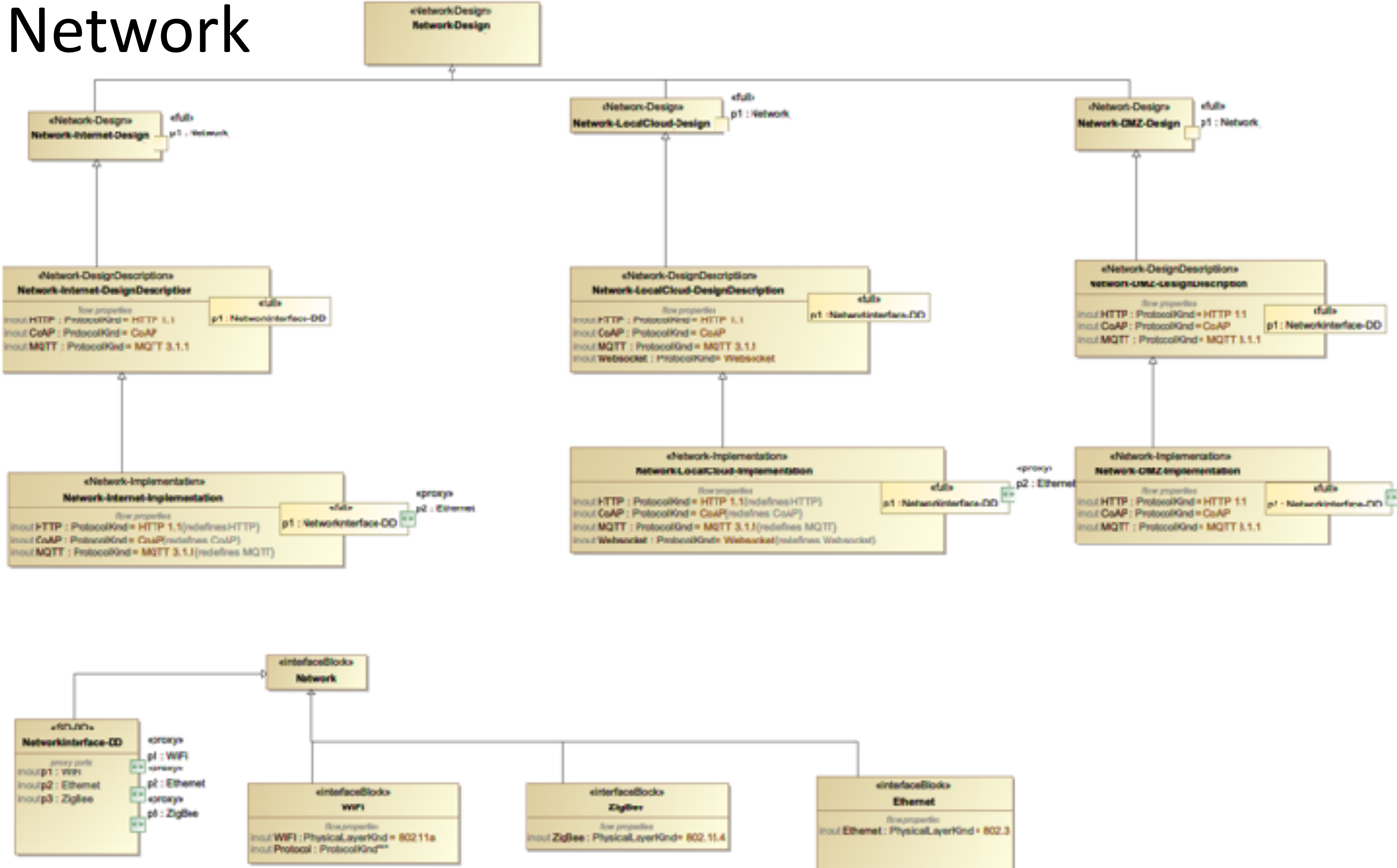
Devices with

- Mandatory core systems
- Support core systems
- Application systems

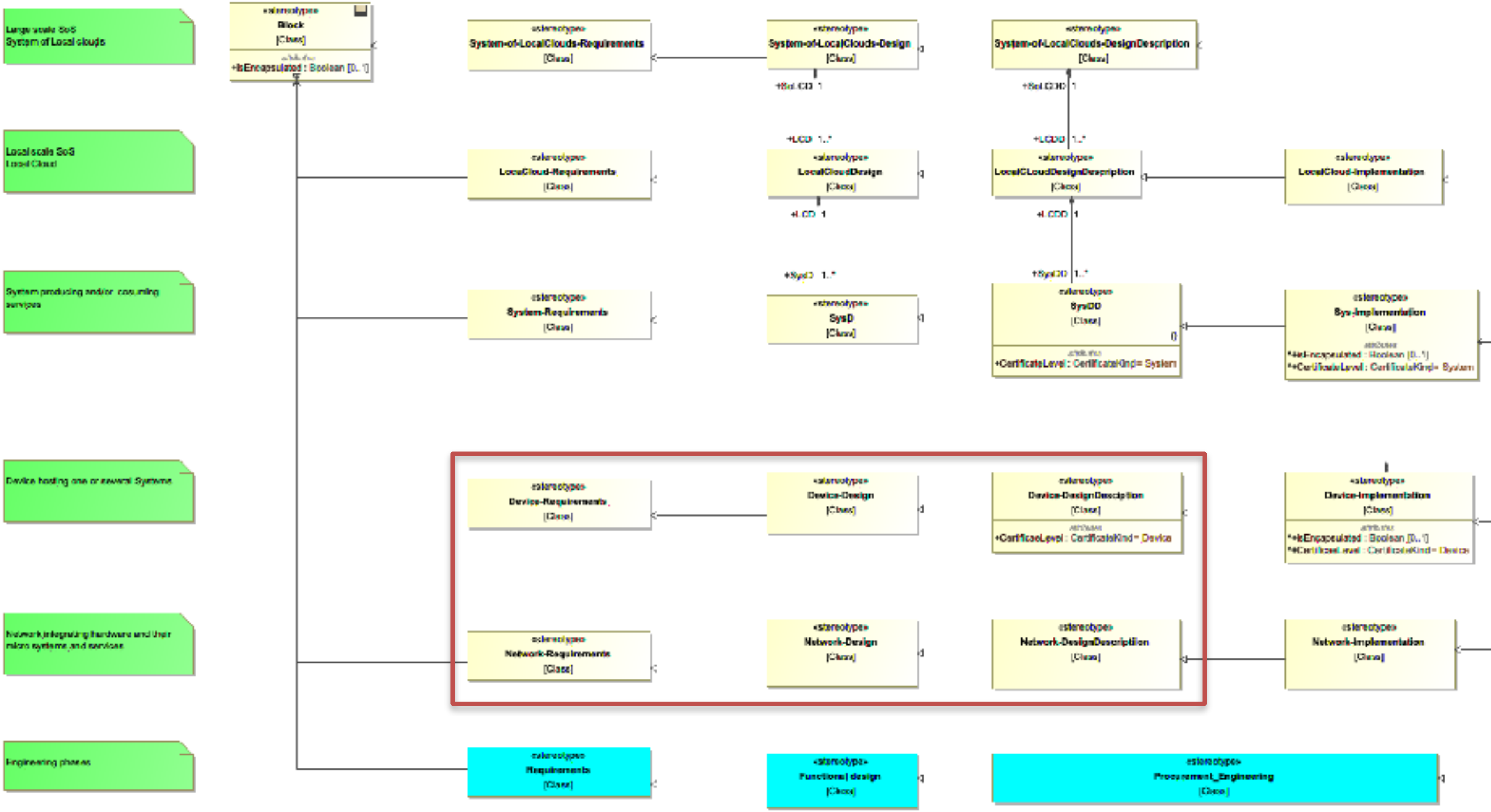
Router



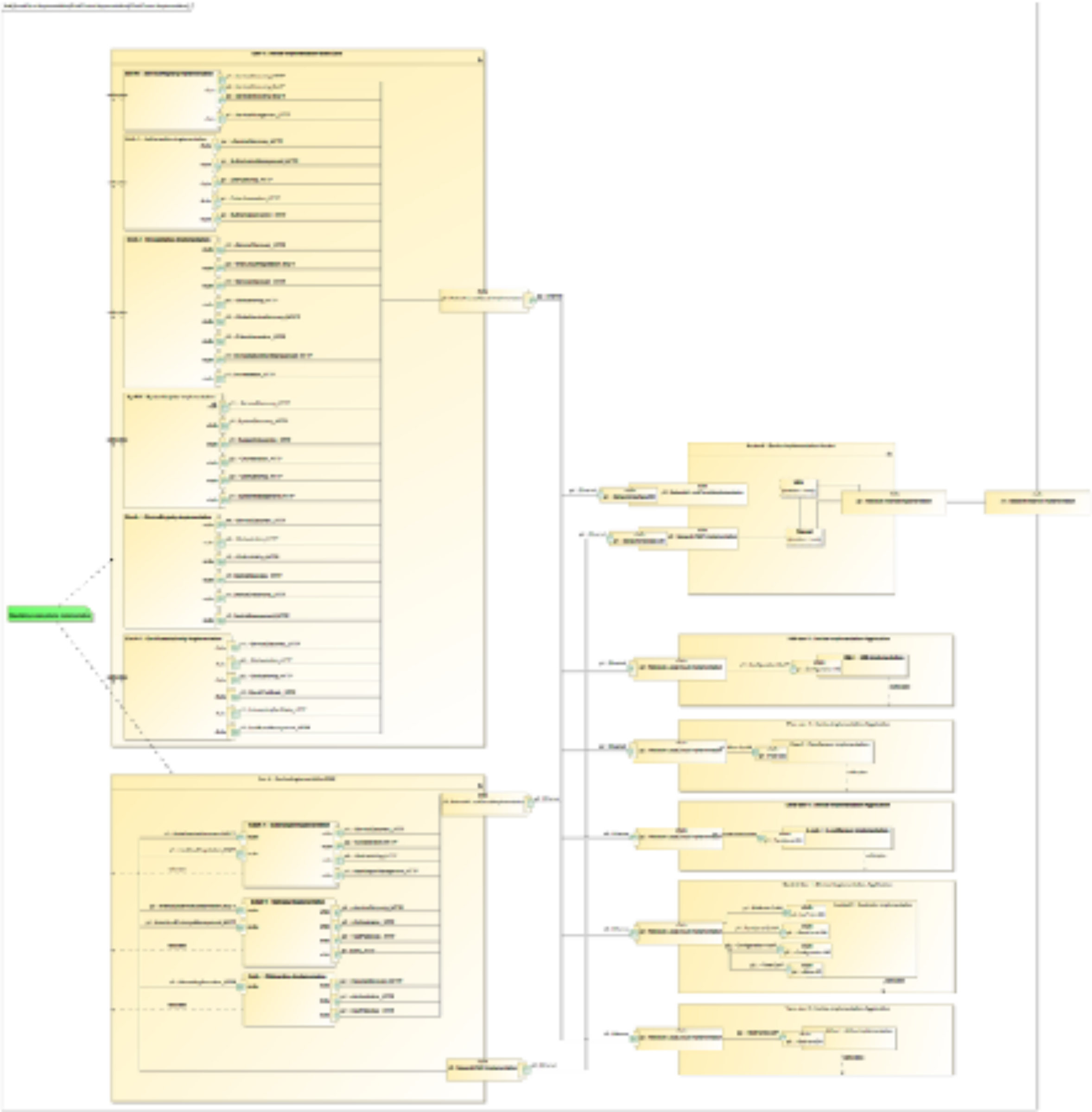
Network


«NetworkDesign»
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-DesignDescription»
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-DesignDescription»
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-DesignDescription»
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»
p1 : Ethernet
«proxy»
p1 : ZigBee
«interfaceBlock»
WiFi
flow properties
inout WIFI : PhysicalLayerKind = 802.11a
inout Protocol : ProtocolKind***
«interfaceBlock»
ZigBee
flow properties
inout ZigBee : PhysicalLayerKind= 802.11.4
«interfaceBlock»
Ethernet
flow properties
inout Ethernet : PhysicalLayerKind = 802.3

SoS architecture and engineering in SysML

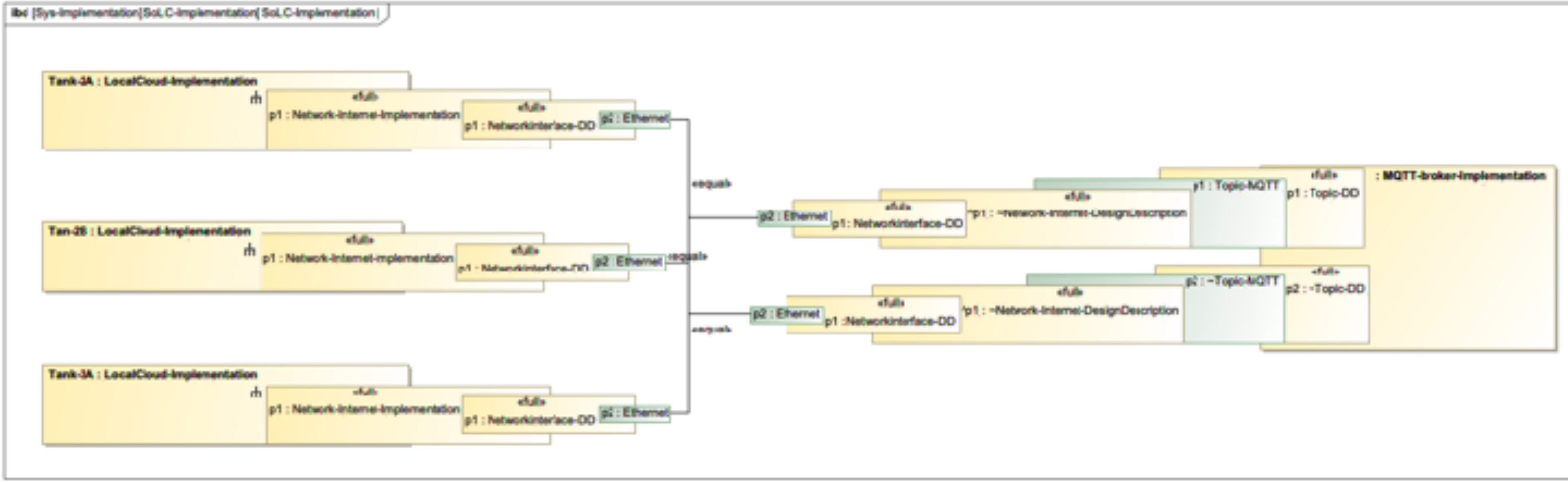


Functional SoS/ Local cloud implementation engineering

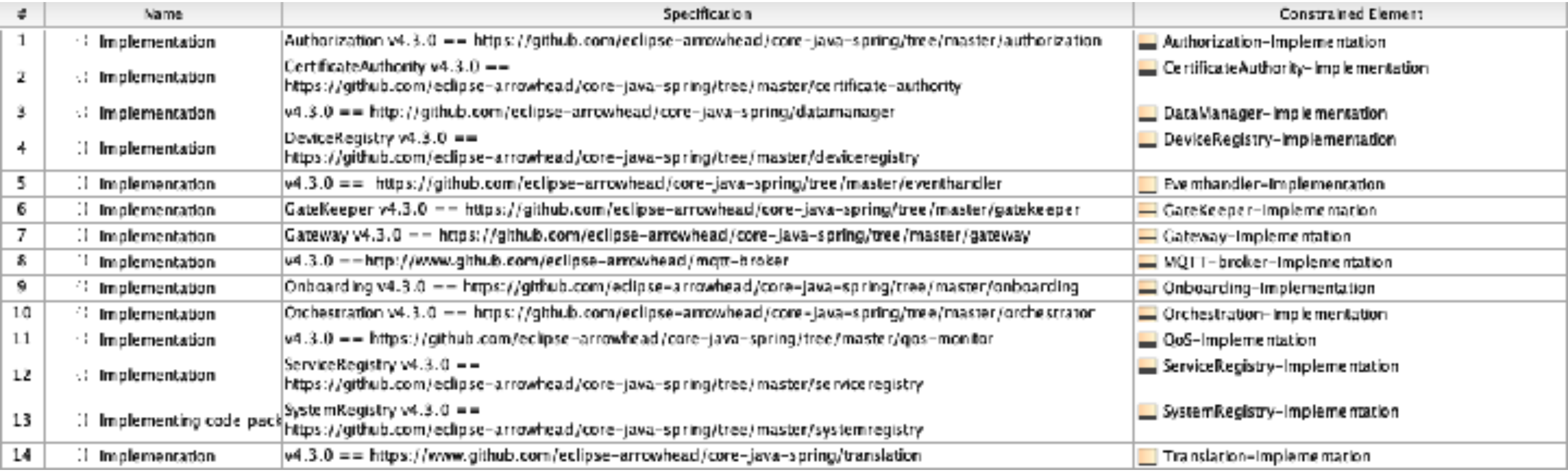


Functional SoLC implementation engineering

System o local clouds functional implementation model


ibd [Sys-Implementation]SoLC-Implementation[SoLC-Implementation]
Tank-3A : LocalCloud-Implementation
«full»
p1 : Network-Interne-Implementation
«full»
p1 : NetworkInterface-DD
p2 : Ethernet
Tan-2B : LocalCloud-Implementation
«full»
p1 : Network-Internet-mplementation
«full»
p1 : NetworkInterface-DD
p2 : Ethernet
Tank-3A : LocalCloud-Implementation
«full»
p1 : Network-Interne-Implementation
«full»
p1 : NetworkInterface-DD
p2 : Ethernet
«equal»
«equal»
«equal»
p2 : Ethernet
«full»
p1: NetworkInterface-DD
«full»
p1 : ~Network-Internet-DesignDescription
p1 : Topic-MQTT
«full»
p1 : Topic-DD
: MQTT-broker-Implementation
p2 : Ethernet
«full»
p1 :NetworkInterface-DD
«full»
p1 : ~Network-Internet-DesignDescription
p2 : ~Topic-MQTT
«full»
p2 : ~Topic-DD

Extraction of code

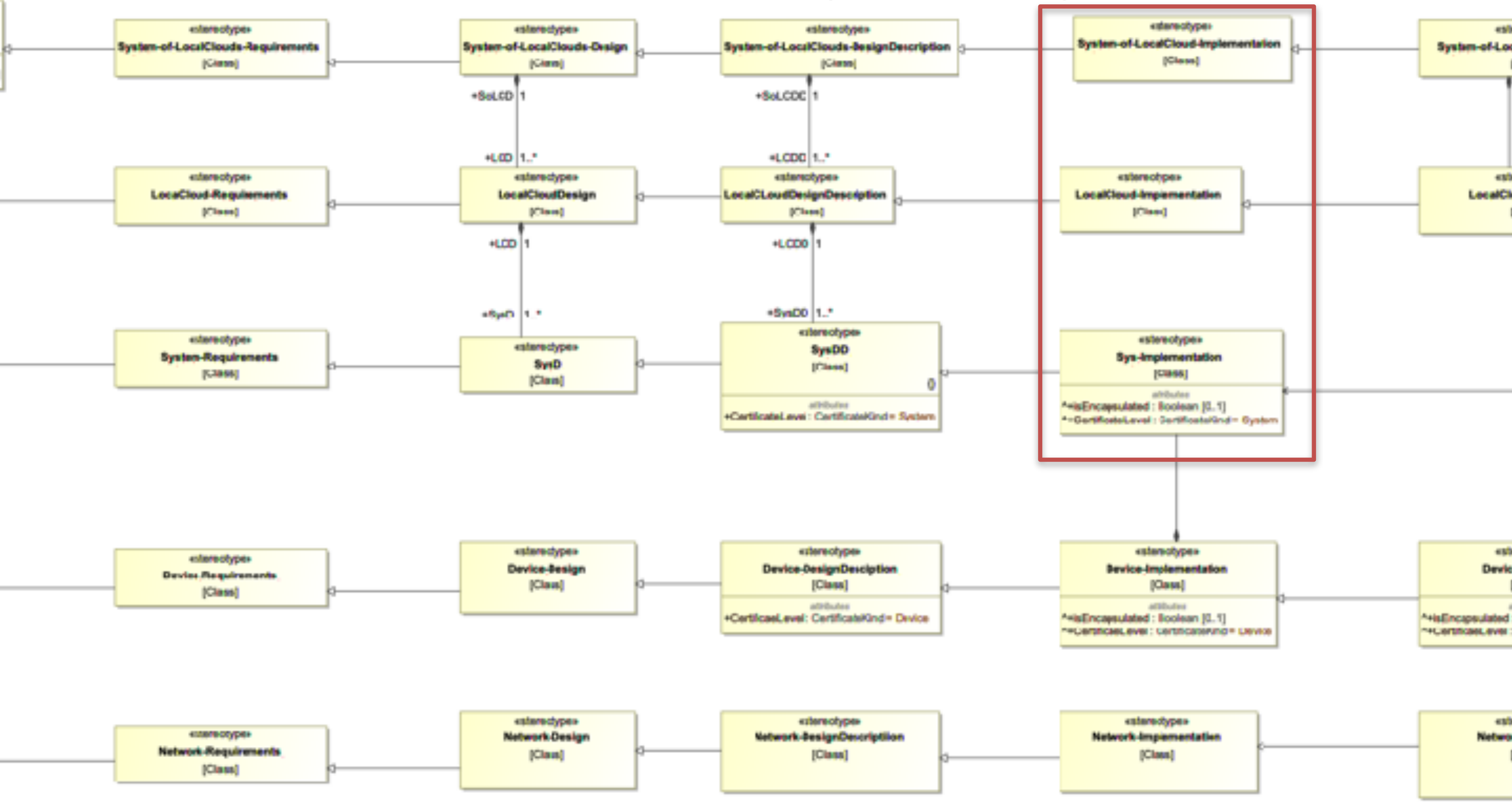


#	Name	Specification	Constrained Element
1	Implementation	Authorization v4.3.0 == https://github.com/eclipse-arrowhead/core-java-spring/tree/master/authorization	Authorization-Implementation
2	Implementation	CertificateAuthority v4.3.0 == https://github.com/eclipse-arrowhead/core-java-spring/tree/master/certificate-authority	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	Orchestration 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

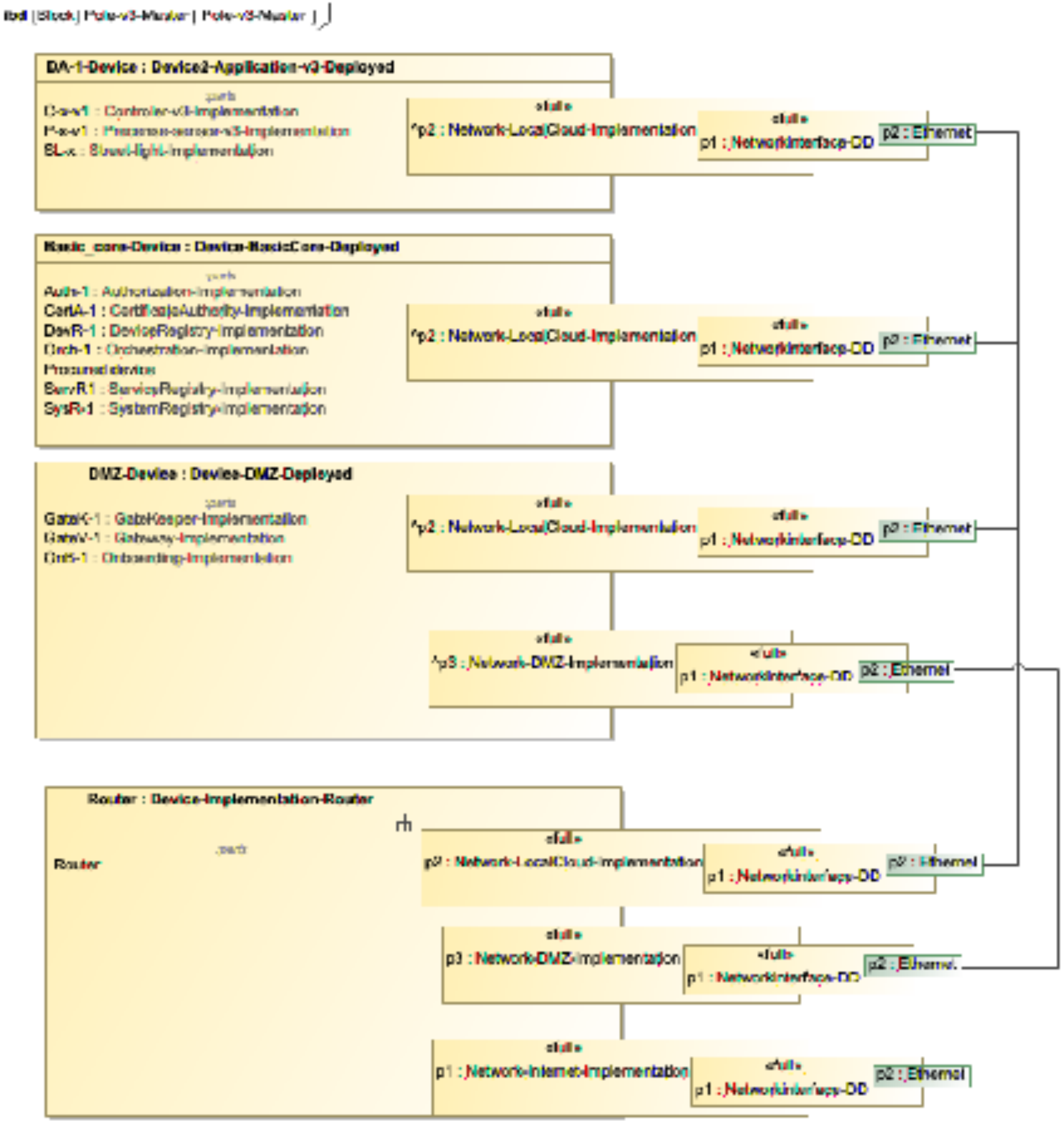
- **Selected HW and OS**
  - **Server - Linux - Ubuntu 20.10**
  - **Desktop computer, Windows 10.xx, OSX 11.2.1**
  - **Embedded system**
    - **Raspberry PI**

SoS architecture and engineering in SysML

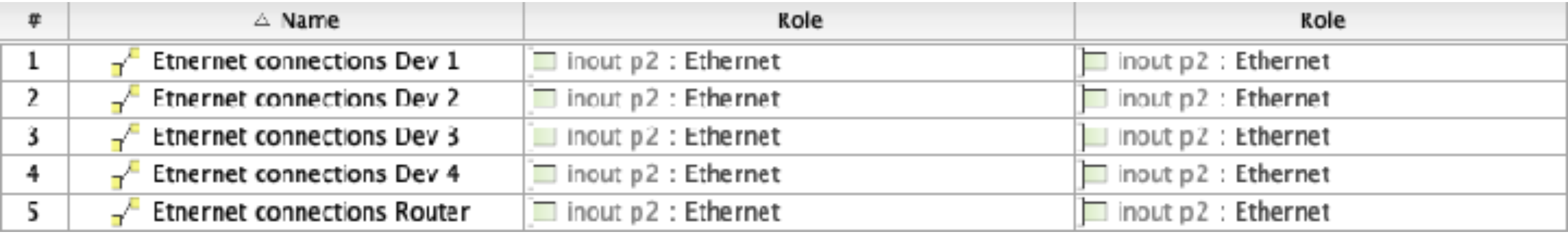


Deployment engineering

SoS/Local cloud implementation engineering


ibd [Block] Pole-v3-Master [Pole-v3-Master]
DA-1-Device : Device2-Application-v3-Deployed
parts
C-a-v1 : Controller-v3-Implementation
P-s-v1 : Presence-sensor-v3-Implementation
SL-x : Street-light-Implementation
«full»
^p2 : Network-LocalCloud-Implementation
«full»
p1 : NetworkInterface-DD
p2 : Ethernet
Basic_core-Device : Device-BasicCore-Deployed
parts
Auth-1 : Authorization-Implementation
CertA-1 : CertificateAuthority-Implementation
DevR-1 : DeviceRegistry-Implementation
Orch-1 : Orchestration-Implementation
Procured device
ServR1 : ServiceRegistry-Implementation
SysR-1 : SystemRegistry-Implementation
«full»
^p2 : Network-LocalCloud-Implementation
«full»
p1 : NetworkInterface-DD
p2 : Ethernet
DMZ-Device : Device-DMZ-Deployed
parts
GateK-1 : GateKeeper-Implementation
GateW-1 : Gateway-Implementation
OnB-1 : Onboarding-Implementation
«full»
^p2 : Network-LocalCloud-Implementation
«full»
p1 : NetworkInterface-DD
p2 : Ethernet
«full»
^p3 : Network-DMZ-Implementation
«full»
p1 : NetworkInterface-DD
p2 : Ethernet
Router : Device-Implementation-Router
parts
Router
«full»
p2 : Network-LocalCloud-Implementation
«full»
p1 : NetworkInterface-DD
p2 : Ethernet
«full»
p3 : Network-DMZ-Implementation
«full»
p1 : NetworkInterface-DD
p2 : Ethernet
«full»
p1 : Network-Internet-Implementation
«full»
p1 : NetworkInterface-DD
p2 : Ethernet

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

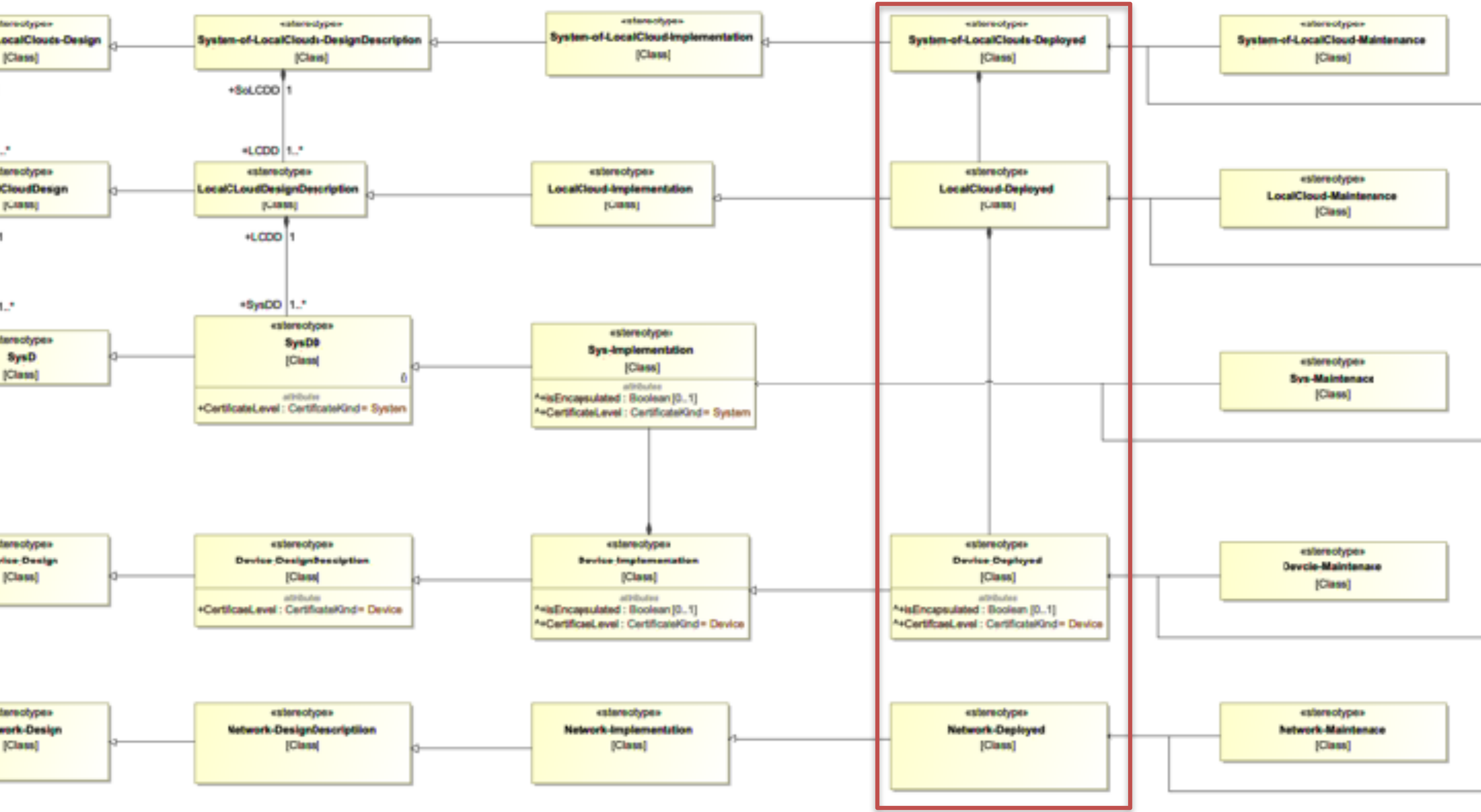
} Orchestration system

Authorisation rules

- PlantDescription system
- Management system

} Authorisation system

SoS architecture and engineering in SysML



Adding functionality

Making use of

- Support core systems models
  - Translator
  - DataManager
  - TimeManger
  - …
- Adaptor systems models
  - OPC-UA -> Arrowhead
  - Modbus TCP -> Arrowhead
  - Z-wave -> Arrowhead
- Application function systems models
  - Code generation from models to executable code

Engineering automation

Move from SysML models of complex SoS

to Docker containers for deployment to

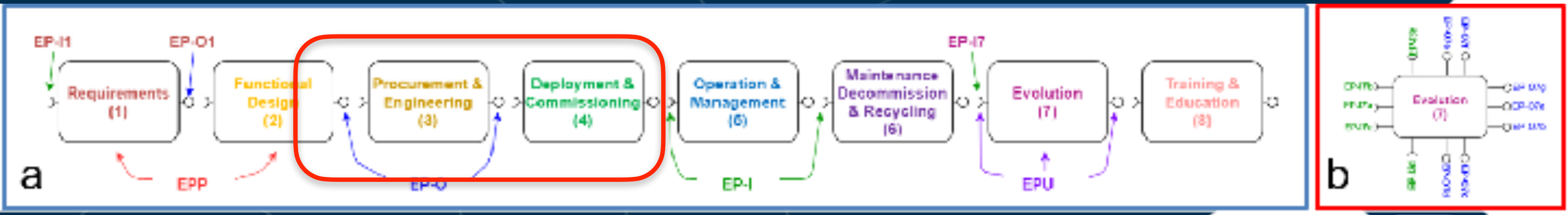
Selected HW and OS

Desktop computer,

Embedded system e.g.

SoS solution generation

- From SysML model of complex SoS
  - Integration with Eclipse IDE
    - Plug-ins
    - Code generation
  - Output
    - Containers of working code
    - Deployable code to selected hardware devices and physical network


EP-I1
EP-O1
Requirements (1)
Functional Design (2)
Procurement & Engineering (3)
Deployment & Commissioning (4)
Operation & Management (5)
Maintenance Decommission & Recycling (6)
EP-I7
Evolution (7)
Training & Education (8)
a
EPP
EP-O
EP-I
EPU
b
Evolution (7)

Conclusions

- SoS solution will rapidly become very complex
- MBE is a time and cost effective approach
  - Automating SoS solution code creation and code reuse
  - Automated extraction of orchestration and security management policies
- Based on open source Eclipse
  - architecture
  - integration framework
  - code and
  - tools
- Github

  www.github.com/eclipse-arrowhead

Availability

Github

www.github.com/eclipse-arrowhead

Comments! Questions?

jerker.delsing@ltu.se