Capella to TASTE MBSE bridge

Model Based Space Systems and Software Engineering - MBSE2020

2020

Agenda

- Project background and objectives
- Implementation
  - Plugin
  - Model
  - Model mapping
  - Workflow
- Validation
  - MTM
  - Lessons Learned

BACKGROUND AND OBJECTIVES

TASTE by ESA

- "A tool-chain targeting heterogeneous embedded systems, using a model-based development approach"
- System architecture – **AADL**
- Data – **ASN.1** (with ACN)
- Behaviour – **SDL** (but can also be **C**, **Ada**...)
- Focused on **software** (but can accommodate FPGAs)
- **Can generate executables** targeting x86, SPARC, ARM and MSP430
- https://taste.tools and https://taste.tuxfamily.org/wiki/

Capella by the Eclipse Foundation

- "A comprehensive, extensible and field-proven **MBSE tool and method** to successfully design systems architecture"
- Based on **Eclipse** IDE (highly customizable)
- Built around the **Arcadia** method
- Provides a layered model to capture **user needs** and **requirements**, perform **system analysis** and **design** a solution
  - User interacts with the model through views
- Cross-domain (both **software** and **hardware**)
- **Cannot generate code** (by itself alone)
- https://www.eclipse.org/capella/

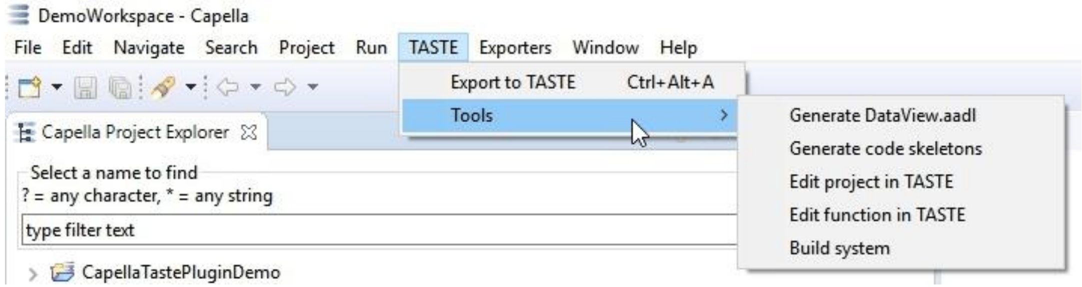
The consortium and the goal

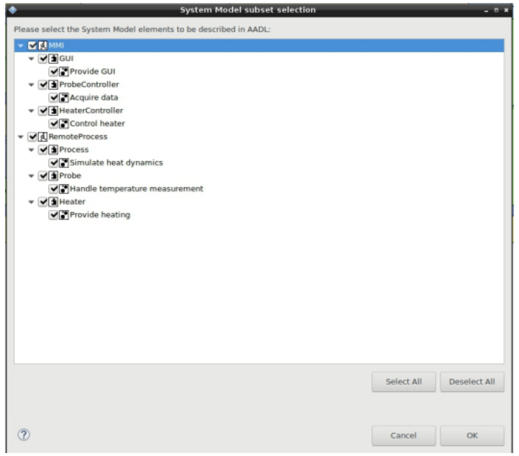
- The **Capella to TASTE MBSE bridge** development was a sub-activity within **MBSE Implement** project founded by **ESA**
- The consortium consisted of:
  - **Creotech Instruments** (prime contractor)
  - **N7 Space** (subcontractor)
- The goal of the sub-activity was to develop a bridge that would allow to apply an **MBSE based approach** throughout the **entire software product lifecycle**, from high-level cross-domain analysis to implementation, testing and deployment

IMPLEMENTATION

Plugin

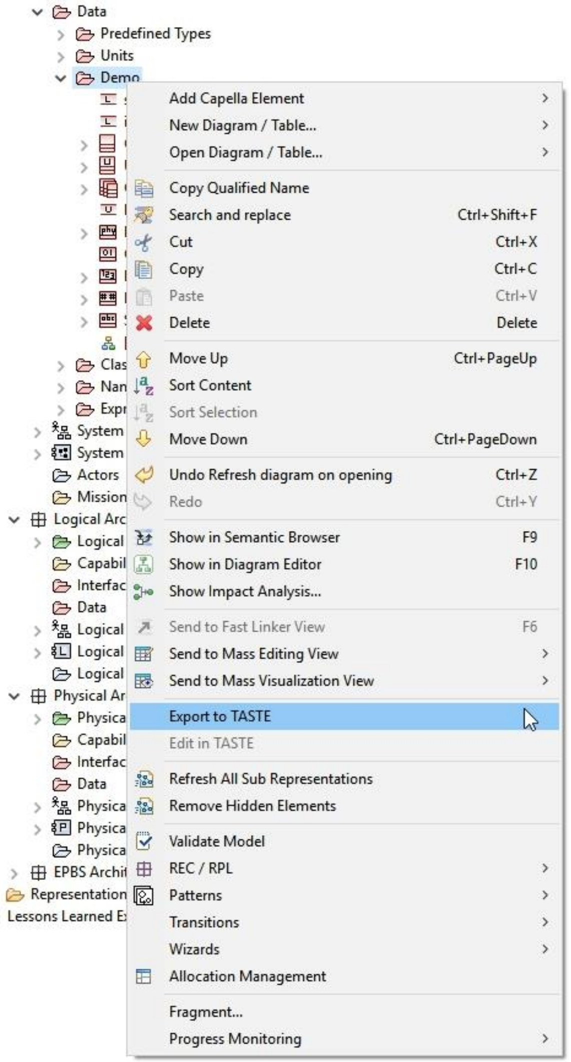
- The bridge is realized as a Capella (Eclipse) plugin
  - Written in Java
- Seamless integration:
  - Context menus
  - Toolbar (with additional TASTE actions)
  - Dialogs
  - Standard preferences





Please select the System Model elements to be described in AADL:

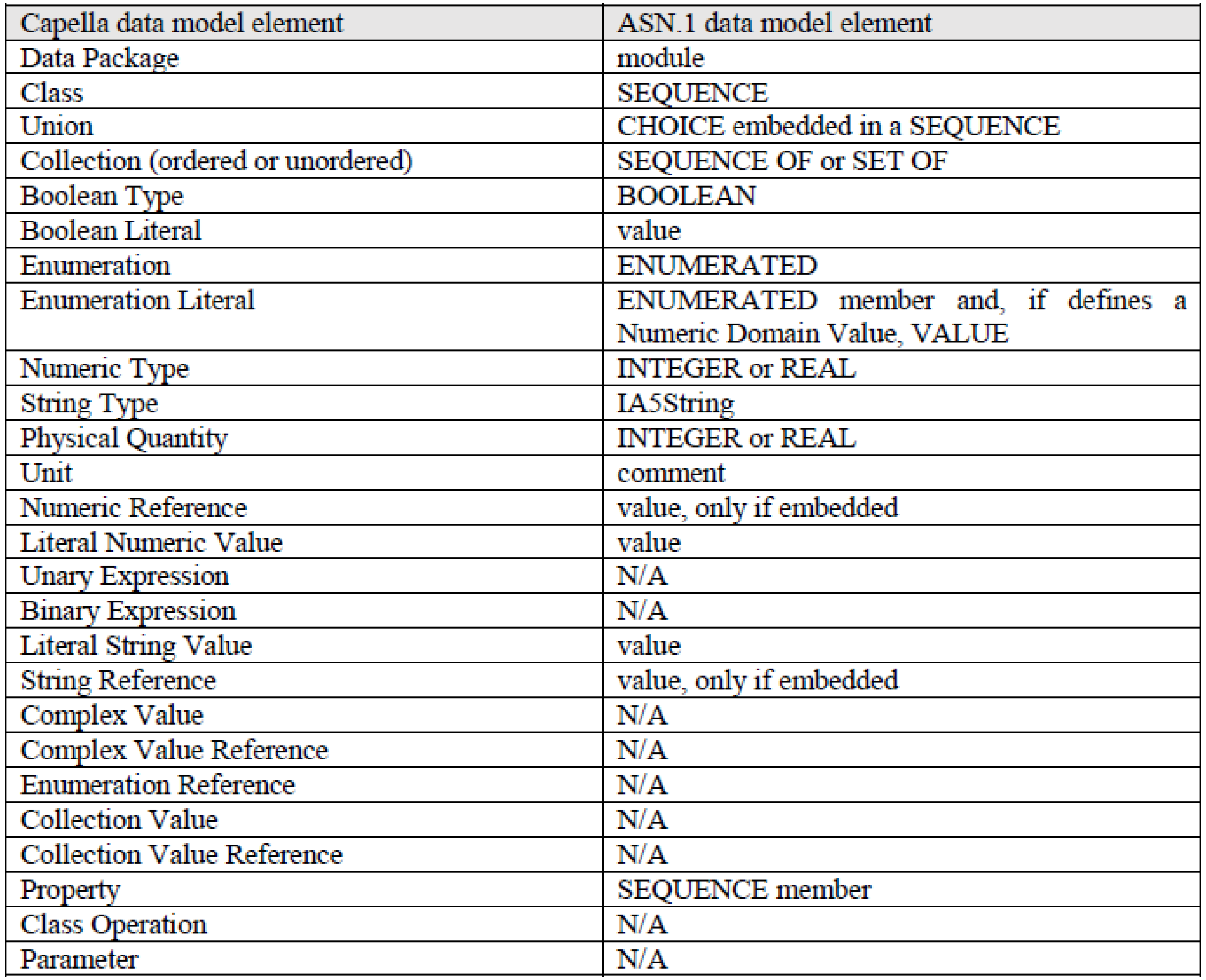
- ☑ MMI
  - ☑ GUI
    - ☑ Provide GUI
  - ☑ ProbeController
    - ☑ Acquire data
  - ☑ HeaterController
    - ☑ Control heater
- ☑ RemoteProcess
  - ☑ Process
    - ☑ Simulate heat dynamics
  - ☑ Probe
    - ☑ Handle temperature measurement
  - ☑ Heater
    - ☑ Provide heating



Model

- **Capella** is high-level, **abstract**, **TASTE** is mid-to-low-level, **concrete**
- Capella data and architecture is well defined
- Capella behaviour model is too abstract
- Model needs to be **constrained** – certain constructs are **not supported**
  - e.g. unconstrained data sizes, some expressions in the data model, shared data, synchronous inter-node communication...
- Model needs to be **supplemented** – constructs must be **concretized**
  - e.g. implementation language, target processor or device config...
  - Provided through string properties
- The plugin performs checks and produces warnings and errors
  - The final verification is performed by TASTE

Model mapping - data

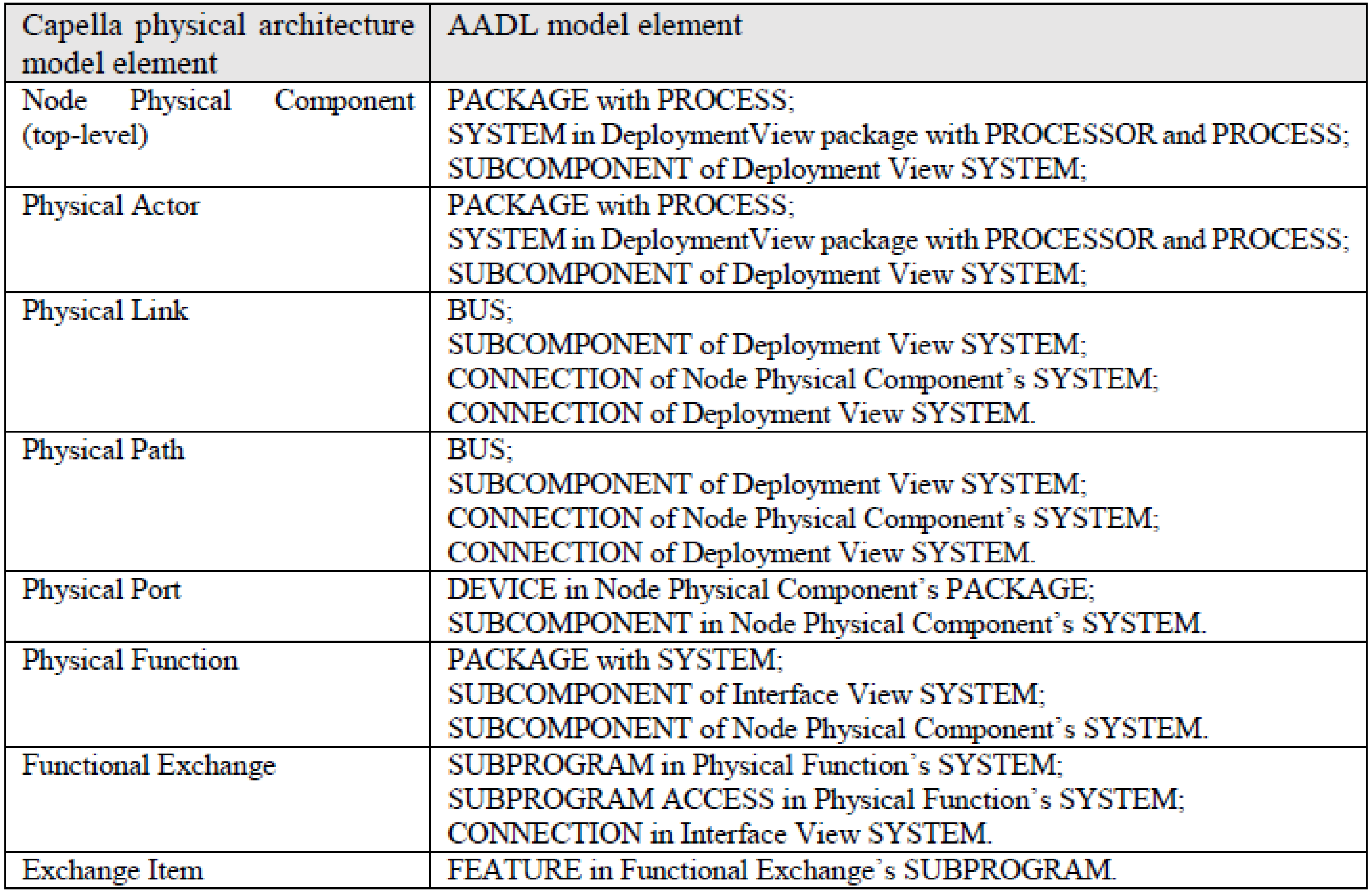


Capella data model element	ASN.1 data model element
Data Package	module
Class	SEQUENCE
Union	CHOICE embedded in a SEQUENCE
Collection (ordered or unordered)	SEQUENCE OF or SET OF
Boolean Type	BOOLEAN
Boolean Literal	value
Enumeration	ENUMERATED
Enumeration Literal	ENUMERATED member and, if defines a Numeric Domain Value, VALUE
Numeric Type	INTEGER or REAL
String Type	IA5String
Physical Quantity	INTEGER or REAL
Unit	comment
Numeric Reference	value, only if embedded
Literal Numeric Value	value
Unary Expression	N/A
Binary Expression	N/A
Literal String Value	value
String Reference	value, only if embedded
Complex Value	N/A
Complex Value Reference	N/A
Enumeration Reference	N/A
Collection Value	N/A
Collection Value Reference	N/A
Property	SEQUENCE member
Class Operation	N/A
Parameter	N/A

*Capella focuses on data semantics, so ACN encoding generation is not supported (can be provided manually); UPER can be used instead

*Some expressions are evaluated during ASN.1 generation

Model mapping - architecture



Capella physical architecture model element	AADL model element

Node Physical Component (top-level)	PACKAGE with PROCESS;
	SYSTEM in DeploymentView package with PROCESSOR and PROCESS;
	SUBCOMPONENT of Deployment View SYSTEM;

Physical Actor	PACKAGE with PROCESS;
	SYSTEM in DeploymentView package with PROCESSOR and PROCESS;
	SUBCOMPONENT of Deployment View SYSTEM;

Physical Link	BUS;
	SUBCOMPONENT of Deployment View SYSTEM;
	CONNECTION of Node Physical Component's SYSTEM;
	CONNECTION of Deployment View SYSTEM.

Physical Path	BUS;
	SUBCOMPONENT of Deployment View SYSTEM;
	CONNECTION of Node Physical Component's SYSTEM;
	CONNECTION of Deployment View SYSTEM.

Physical Port	DEVICE in Node Physical Component's PACKAGE; SUBCOMPONENT in Node Physical Component's SYSTEM.

Physical Function	PACKAGE with SYSTEM;
	SUBCOMPONENT of Interface View SYSTEM;
	SUBCOMPONENT of Node Physical Component's SYSTEM.

Functional Exchange	SUBPROGRAM in Physical Function's SYSTEM;
	SUBPROGRAM ACCESS in Physical Function's SYSTEM;
	CONNECTION in Interface View SYSTEM.

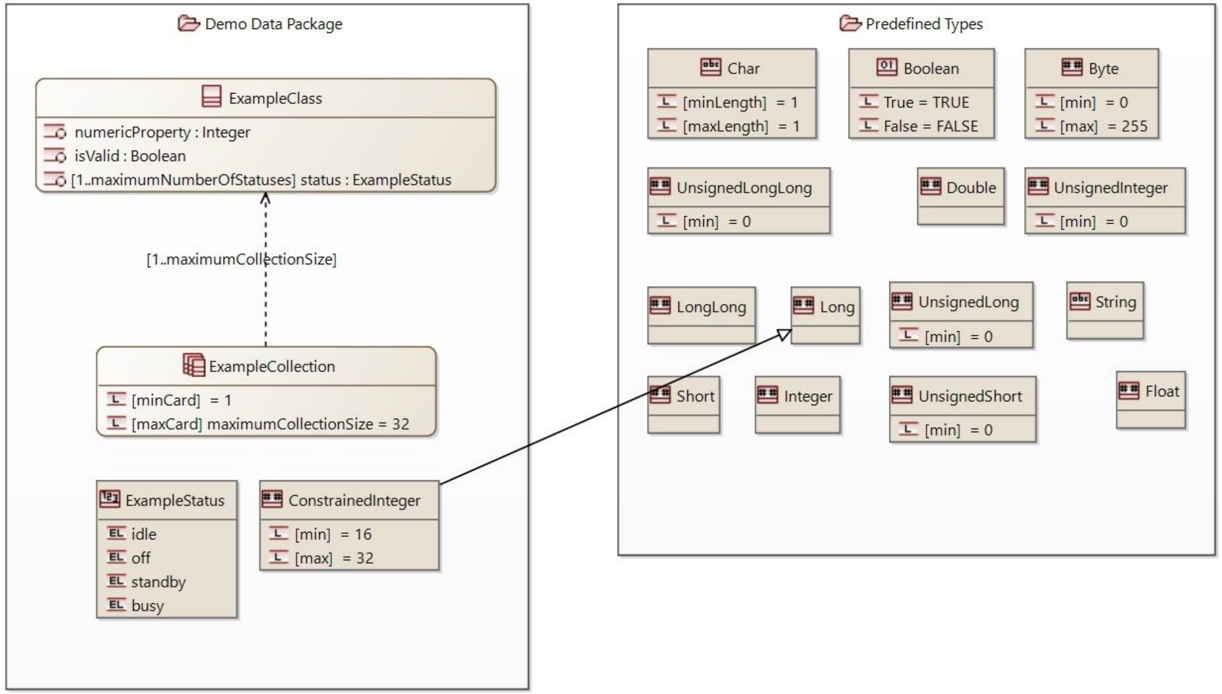
Exchange Item	FEATURE in Functional Exchange's SUBPROGRAM.

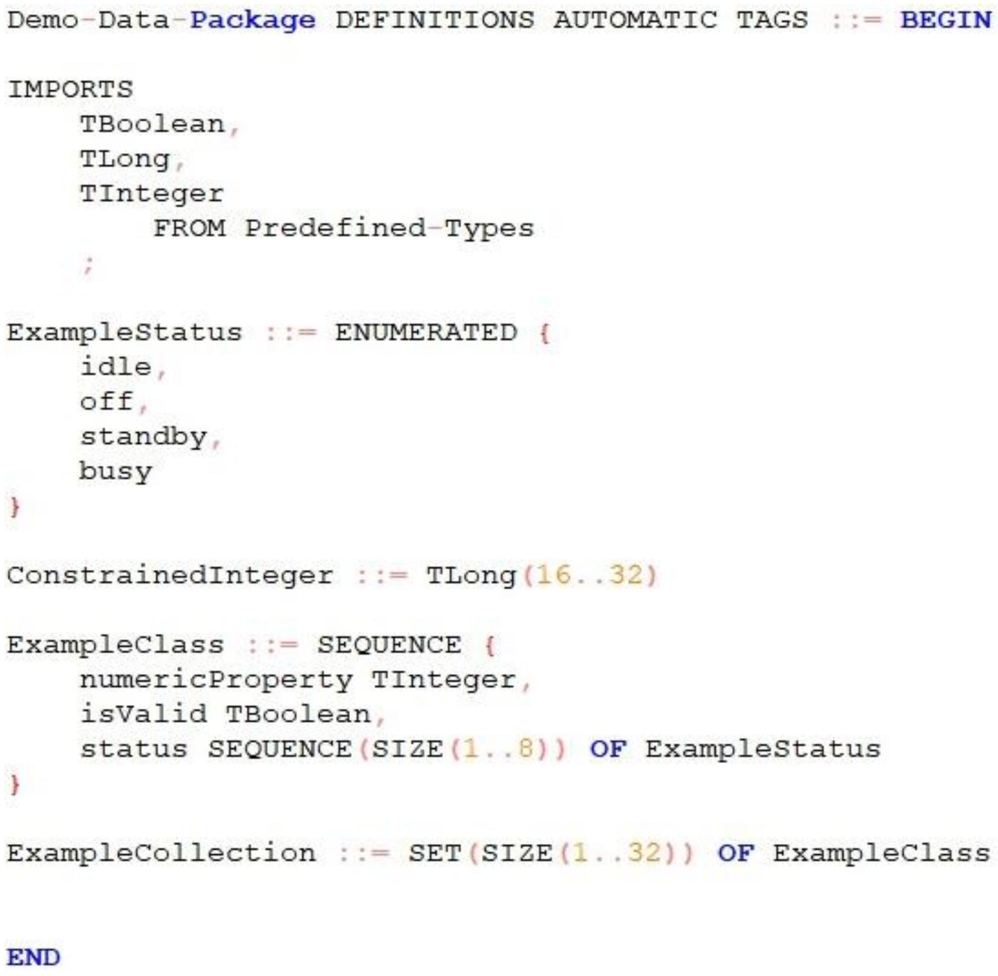
*Functional Exchange is the most complex construct to map between Capella and TASTE

Workflow

- Define the **data model** in Capella
- Define the **architecture** in Capella
- Apply the required properties
- Export (a selection of) the data model to **ASN.1**
- Export (a selection of) the architecture to **AADL**
- Perform post-processing (**ASN.1 -> AADL**, **code skeletons**)
- Define **behaviour** in TASTE
- **Compile, deploy, test**

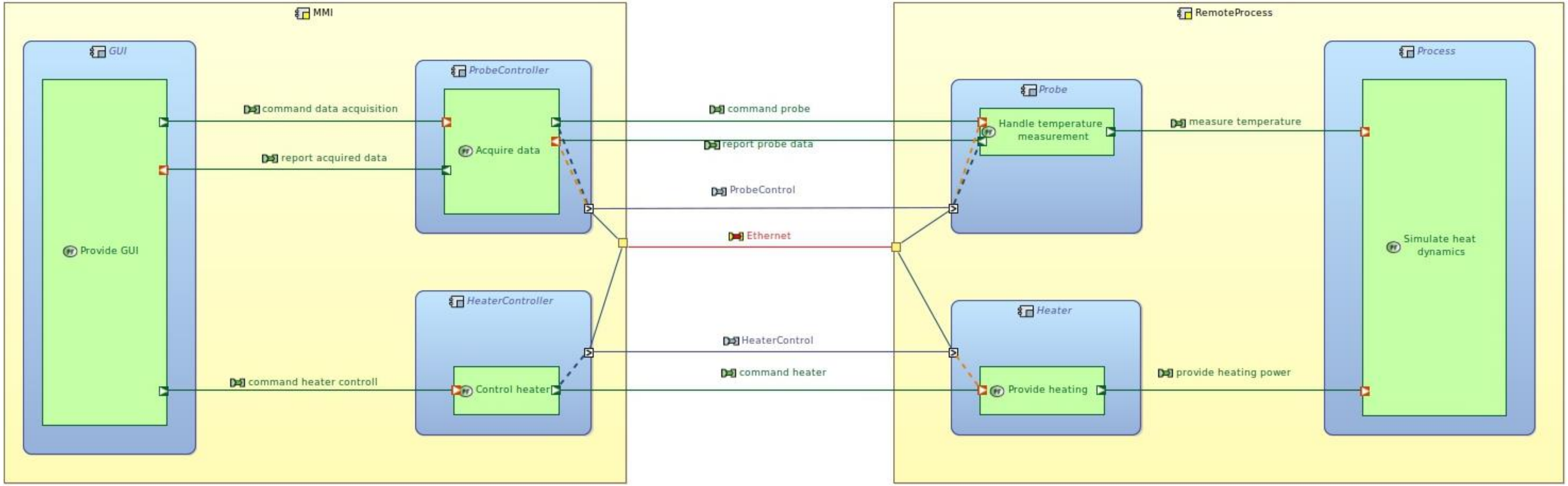
Capella data and TASTE ASN.1



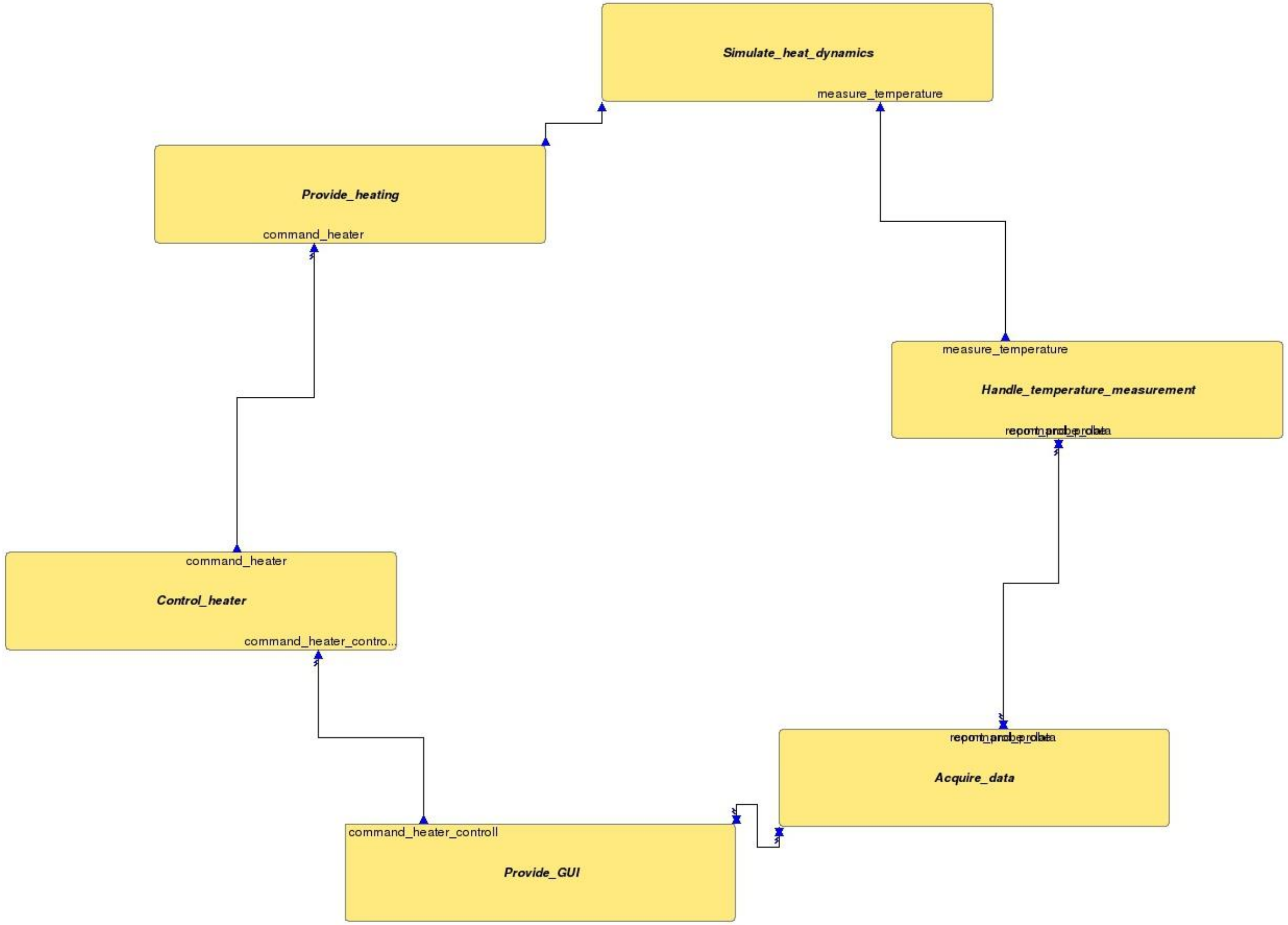


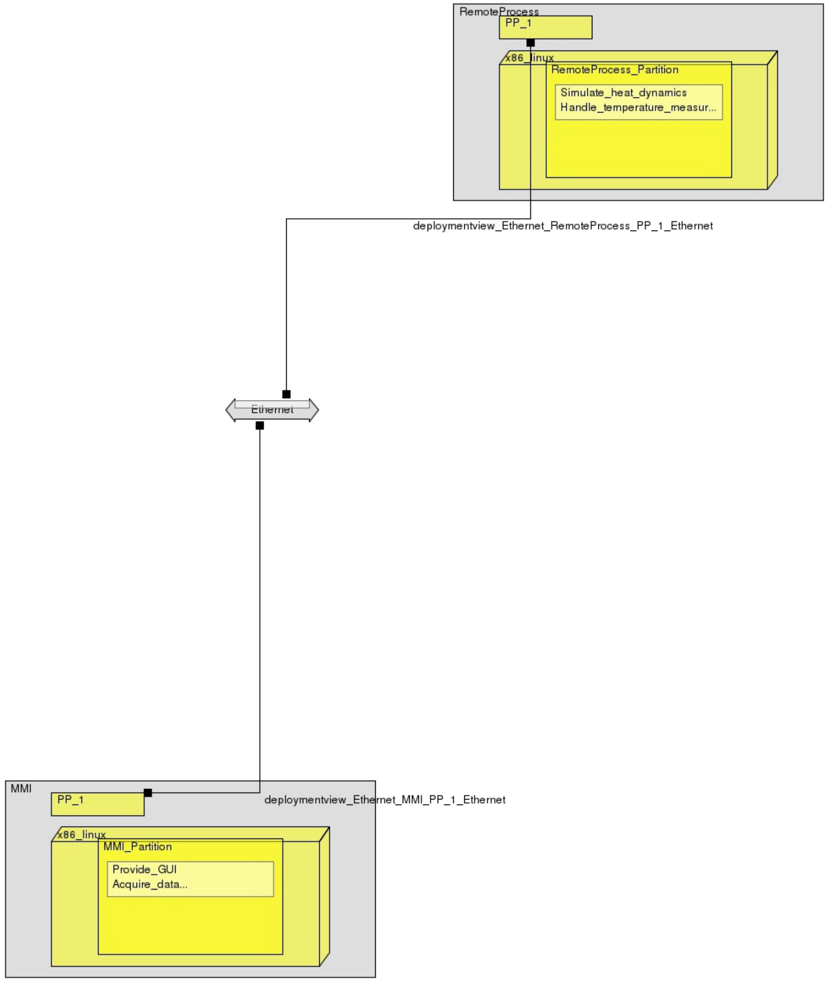
```
Demo-Data-Package DEFINITIONS AUTOMATIC TAGS ::= BEGIN

IMPORTS
    TBoolean,
    TLong,
    TInteger
        FROM Predefined-Types
    ;

ExampleStatus ::= ENUMERATED {
    idle,
    off,
    standby,
    busy
}

ConstrainedInteger ::= TLong(16..32)

ExampleClass ::= SEQUENCE {
    numericProperty TInteger,
    isValid TBoolean,
    status SEQUENCE(SIZE(1..8)) OF ExampleStatus
}

ExampleCollection ::= SET(SIZE(1..32)) OF ExampleClass


END
```

Capella physical architecture



TASTE Interface and Deployment Views





VALIDATION

MTM

- Mass-and-Thermal Mockup, based on STM32F407 MCU (32-bit ARM)
- Hardware and Capella model were developed by Creotech
- Capella model was transformed into TASTE model using the plugin
- Behaviour was modelled in SDL and C by N7 Space
  - C code based on an alternative twin software manually coded by Creotech
- TASTE RS-485 driver was implemented by N7 Space in Ada
- Code was deployed onto the MTM hardware
- Automated test scenarios, defined by Creotech, were implemented in Python by N7 Space based on code auto-generated from MSC diagrams created in TASTE
- Result: everything works!

Lessons Learned

- Environment setup may be non-trivial
- (Pre-Kazoo) Project setup was cumbersome
- (Pre-Kazoo) Project build was slow
- MBSE ensured that the **logic was sound**
  - but there were also the "**small implementation details**" - bugs were found in **drivers**, and **memory corruption** debugging was time consuming
- Memory consumption was a challenge ("only" **128 kB** of accessible RAM)
- Kazoo, introduced when this project was underway, **solved** many of these issues!
- Some of the other issues were solved later in the **Tiny Runtime to Run Model-Based Software on CubeSats** project (a similar complexity fits into **54 kB** of **total** memory)

Lessons Learned

- Naming convention is important
- Inheritance support has limitations
- Sizes must be defined and constrained
- Use built-in strings
- Be aware of the implementation constraints
- Remember that TASTE handles the communication layer on its own
- A single GUI improves usability and makes automated testing much easier
- When behaviour is implemented, iteration is getting expensive

Lessons Learned – the good part

- MBSE makes it easier to reach **an unambiguous understanding** between partners
- MBSE ensures **strict adherence** of the implementation to the design
  - interfaces just match!
- MBSE relieves the implementer from **some** low-level tasks, **potentially** improving **delivery speed and cost**

- **What's next?**
  - **Optimize runtime** (initial work done for MSP430, some other platforms don't need it)
  - Make the tooling more **user-friendly** (in progress)
  - Provide **more drivers, more runtimes**

SUMMARY

Project achievements

- **Capella data and physical architecture** model can be now exported to **TASTE ASN.1 and AADL** models
  - Capella can be used for user needs capture, requirement tracing, system analysis and high-level design
- The **ASN.1** and **AADL** models can be supplemented by behaviour definition (**SDL**, C or Ada...) in **TASTE**
  - TASTE can be used for **code generation** (full or partial)
- **Entire software development cycle is supported by an MBSE based approach**

Thank you for your attention

Michał Kurowski
mkurowski@n7space.com

Michał Mosdorf
mmosdorf@n7space.com

Michał Kocon
michal.kocon@n7space.com

+48 22 299 20 50
www.n7space.com