*A*vionics *S*ystems *H*osted on a distributed modular electronics *L*arge scale d*E*monstrator for multiple t*Y*pe of aircraft

Presented by

Martin Halle (TUHH)

Prepared by

Martin Halle (TUHH)

Configuration Management for IMA Components

ASHLEY-WP73-TUHH-DISM-PRES-0483

EUROPEAN COMMISSION
European Research Area

Preface

This publication only reflects the view of the ASHLEY Consortium or selected participants thereof. Whilst the ASHLEY Consortium has taken steps to ensure that this information is accurate, it may be out of date or incomplete, therefore, neither the ASHLEY Consortium participants nor the European Community are liable for any use that may be made of the information contained herein.

Presentation Content

- *Introduction*
- *Meta-Model for IMA Configuration Data*
- *Platform Configuration Tool*
- *Transition between tools: Model2Model*
- *Summary and Outlook*
- *Contacts*

Introduction:
TUHH/FST Institution – Overview

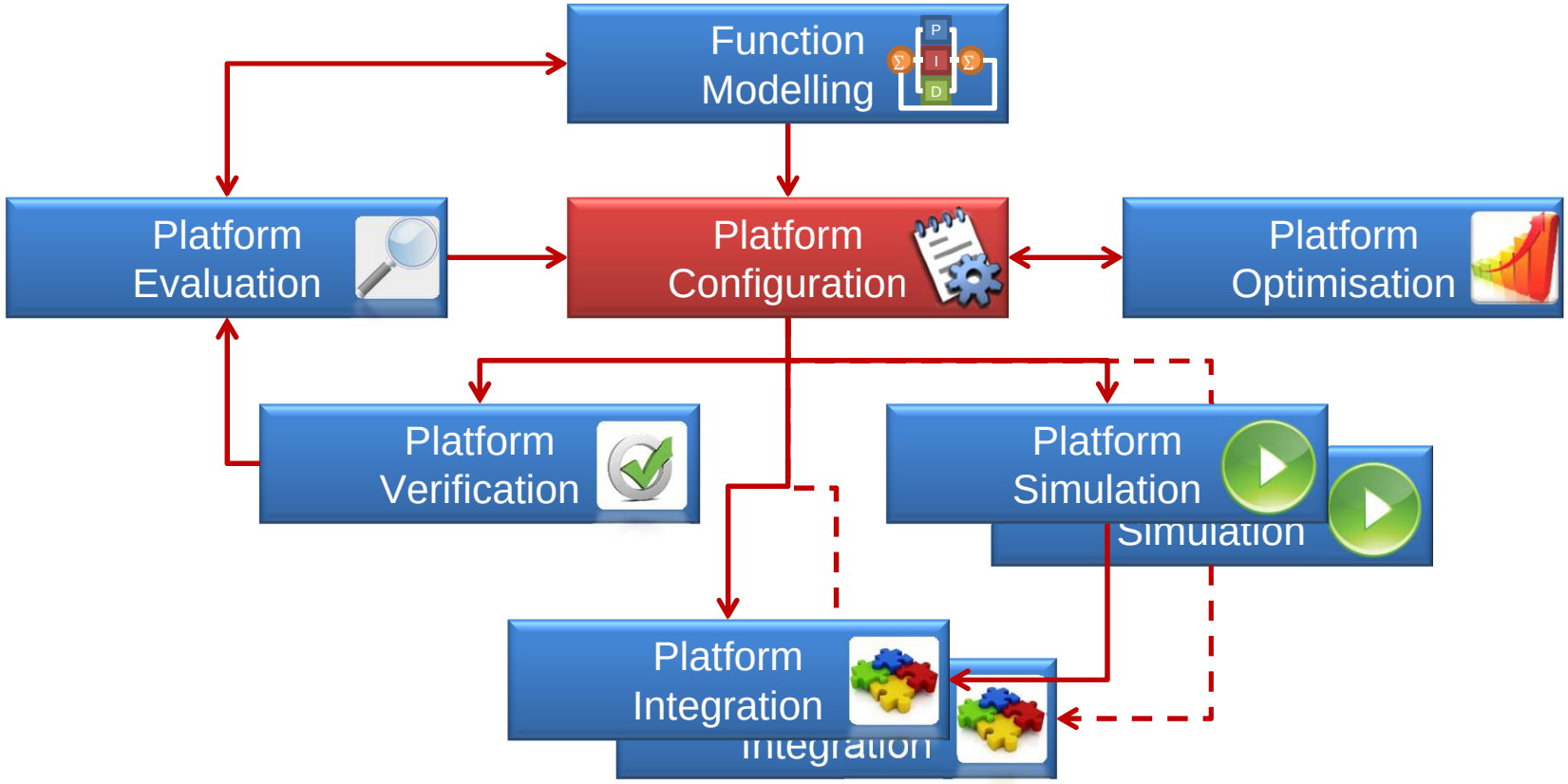
- Hamburg, University of Technology – Institute of Aircraft Systems Engineering
- Situated in Hamburg/Germany, next to Airbus (sharing the building)

- Head of department, secretary, 6 technicians, 25 research engineers
- Different research groups in areas like hydraulics, electrics, cooling, *avionics*

- Research facility for large test benches
- Technology demonstration for different aircraft systems in cooperation with industrial partners

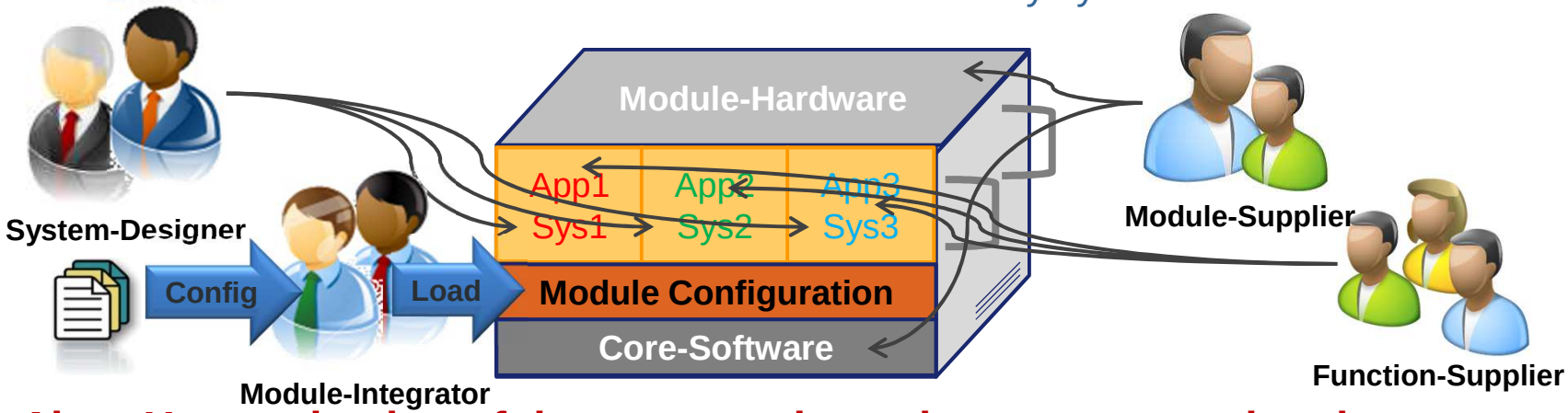
Introduction:
TUHH/FST @ ASHLEY

- Goal within ASHLEY:
  - Harmonise IMA development process with other tools "*seamless tool chain*"
  - Develop and enhance interfaces and tools for IMA2G
  - Support configuration process for demonstrator



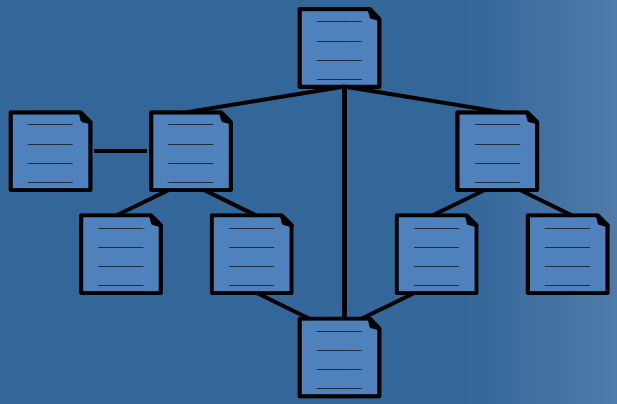
Introduction:
The IMA Configuration Management Process

- Challenges
  - Many thousands of configuration parameters need to be managed
  - Hard- and software development phase (new A/C programs):
    - ✓ Hard the define/specify exact resource needs of systems and their applications
      → Development process in iterations
    - ✓ What parameters exactly and how they need to be configured may not be known
      → Parallel development of hard- and software
- Integration phase:
  - Distributed process: Different actors specify different parts of the configuration
  - Different tool chains for different modules allocated by systems



→ Aim: Harmonisation of the process through computer assisted configuration engineering and validation for IMA modules

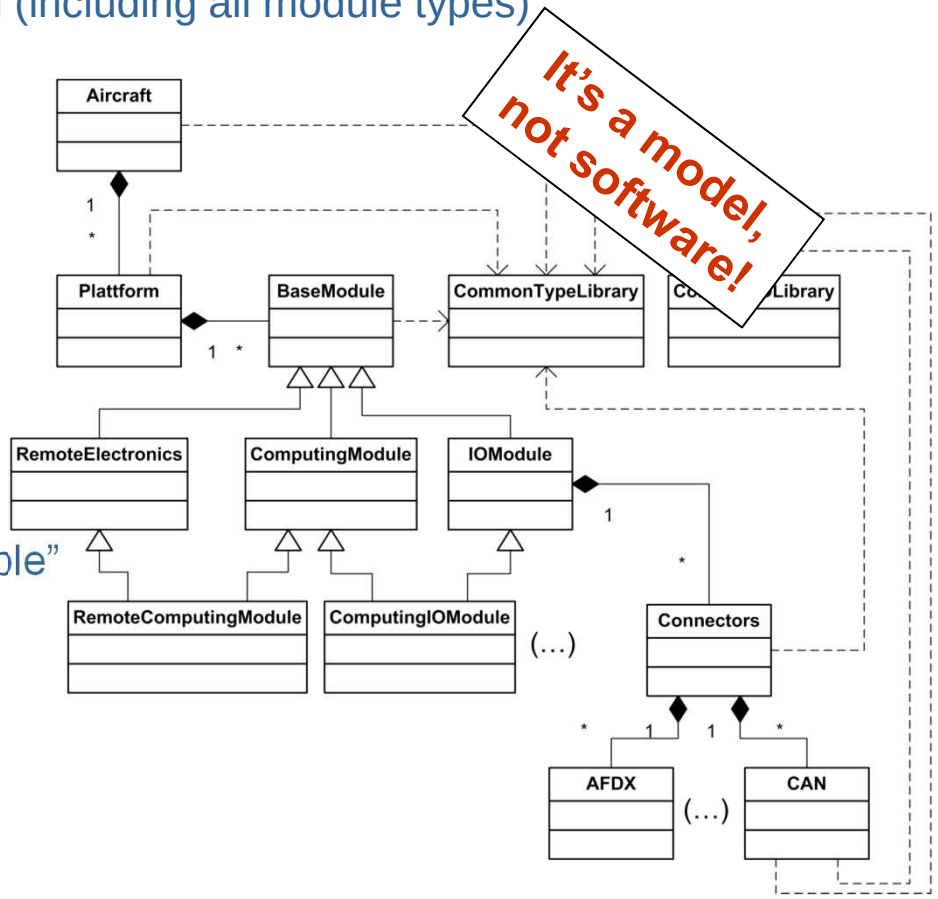
Meta-Model for IMA Configuration Data



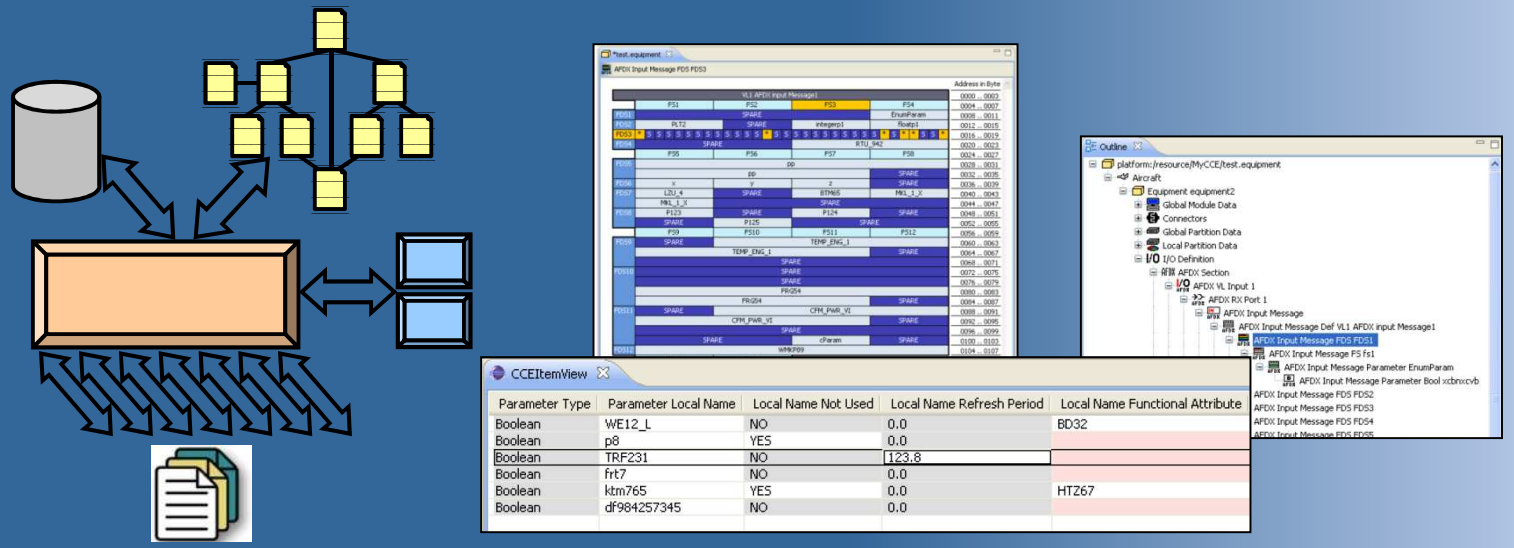
Meta-Model for IMA Configuration Data

- Requirements for the modelling domain
  - Include relevant configuration parameters for dedicated process steps
  - Encapsulate different parts of the configuration (actors)
  - Cover IMA configuration at aircraft level (including all module types)
  - Consider standards and commonalities
  - Module supplier independent
  - Extensible for future needs

- Realisation
  - There is no standard available (A653 part 1 is not sufficient)
  - Development of own Meta-Model
  - Object-oriented approach for a “pluggable” structure (→ configuration tree)
  - Make use of abstractions and libraries
  - Covers different module types and I/O parameters
  - A653 compatible



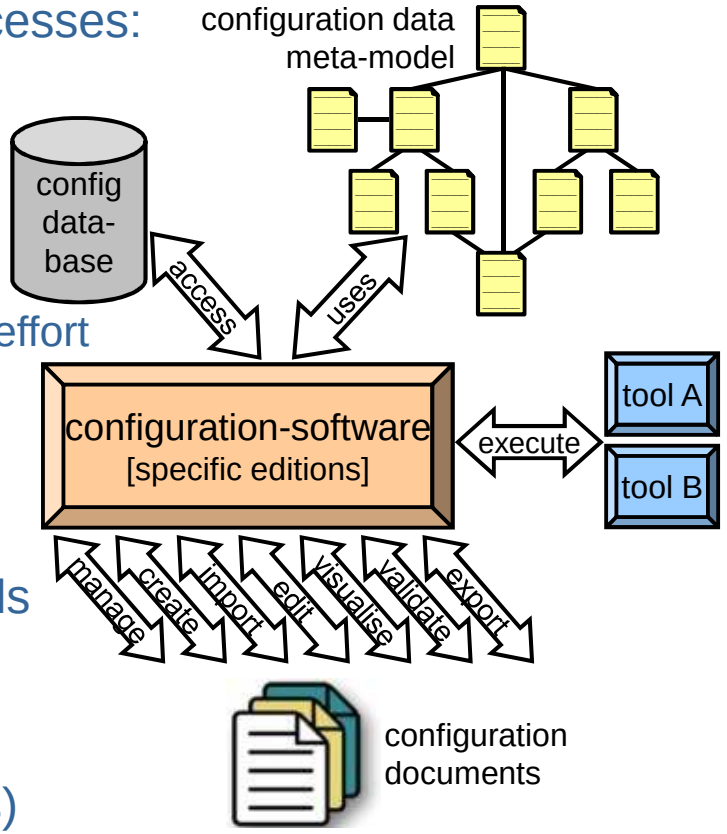
Platform Configuration Tool



Platform Configuration Tool

Rationale and areas of use

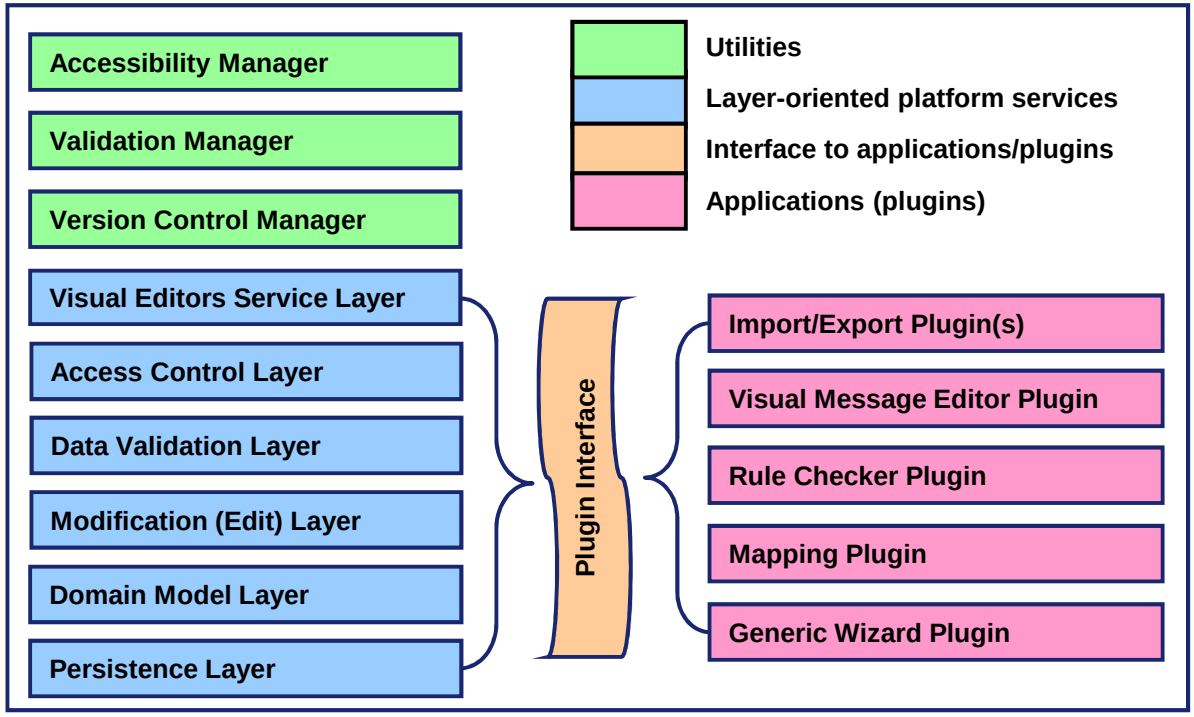
- Target nowadays mostly manual, error-prone processes:
  - Create/manage configuration data at aircraft level
  - Specification, resource negotiations
  - Creation of interface description documents (ICD)
  - Import/export from/into existing formats → re-use
  - Early configuration validation → reduce integration effort
  - Visualise complex relationships
- One integrated instead of many dedicated tools
- Integrate into configuration process with other tools
- Supports mainly Airframer activities
- Supports Function Supplier's processes (not tools)
- No replacement for (qualified) Module Supplier tools



Platform Configuration Tool

Implementation

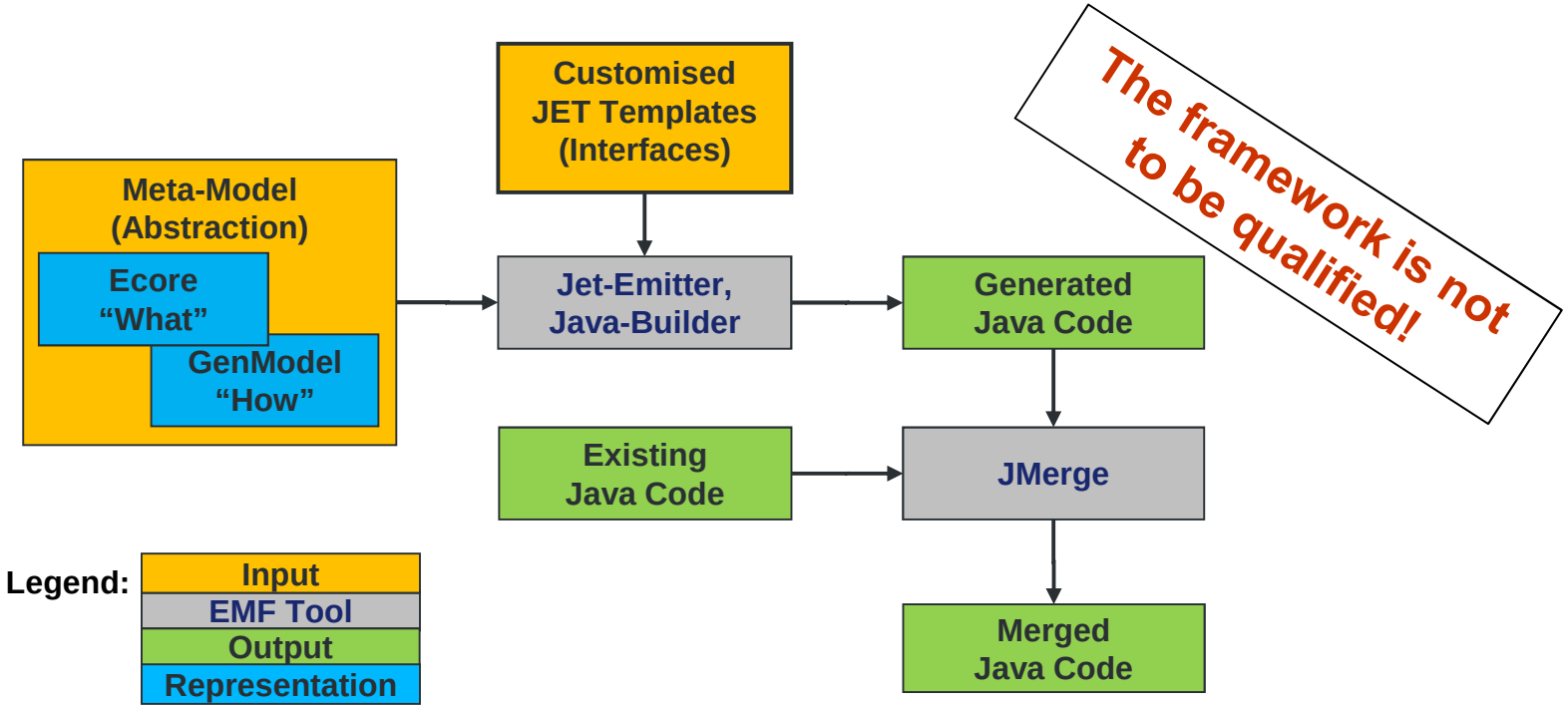
- "*Everything must be flexible.*": A programmers nightmare.
- Integrated software suite based on **plugins** and **extensions** following the **Eclipse** concept
- Heavy use of modern software frameworks
- Layer based architecture
- Interfaces, many of them!
- Functionality via Plugins



Platform Configuration Tool

“Morphable” Software through Code Generation

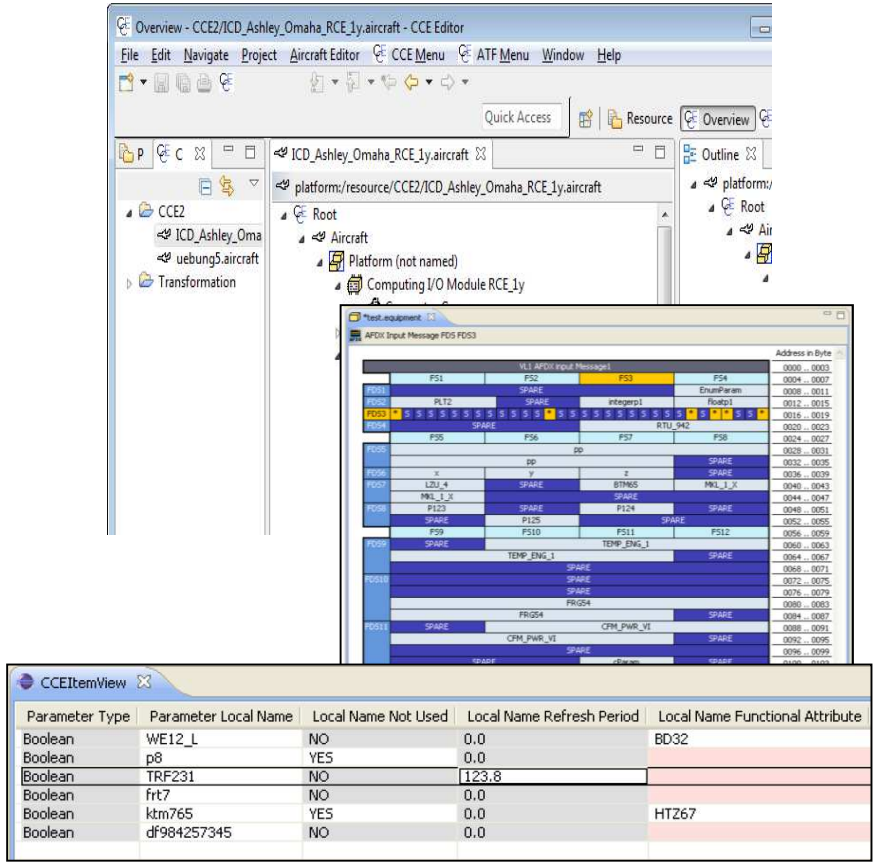
- Based on Eclipse Modelling Framework (EMF)
- Translates Meta-Model into software core (model-based S/W framework)
- Allows for fast adoption cycles
- Allows for code customisations (e.g. interfaces for plugins)
- Creates *all* lower layers of the presented architecture



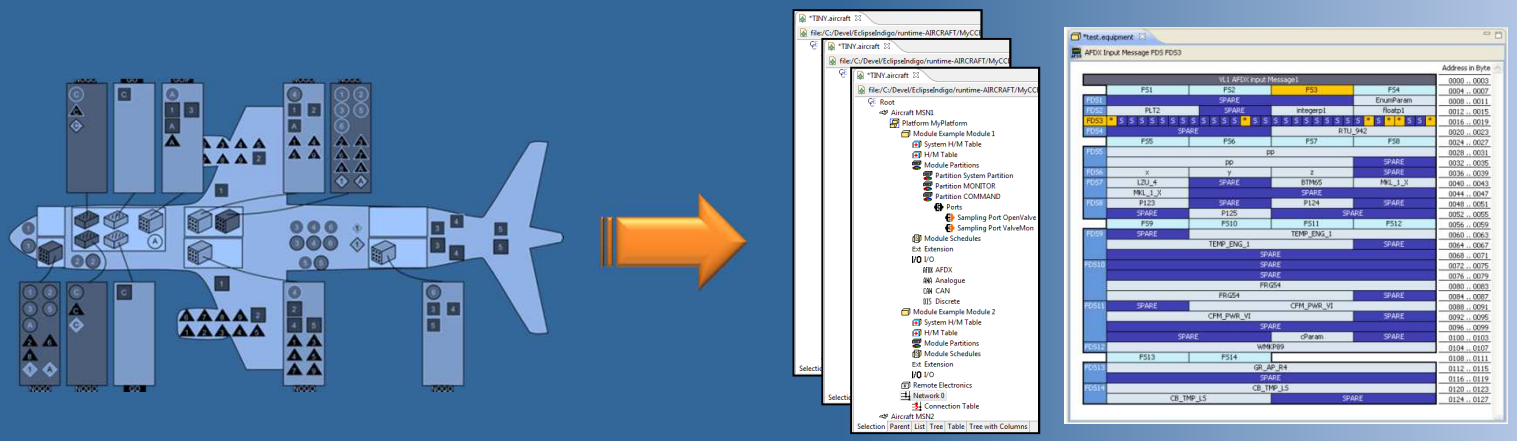
Platform Configuration Tool

Achievements

- Identified data flow and work flow for *enhanced* IMA process
- Developed generic meta-model for Platform Configuration Data for IMA modules used in demonstrator
- Achieved on-the-fly and user-triggered validation of Configuration Data for consistency
- Development of interface for all tools in the “ASHLEY Tool Framework”
- Achieved different visual representations



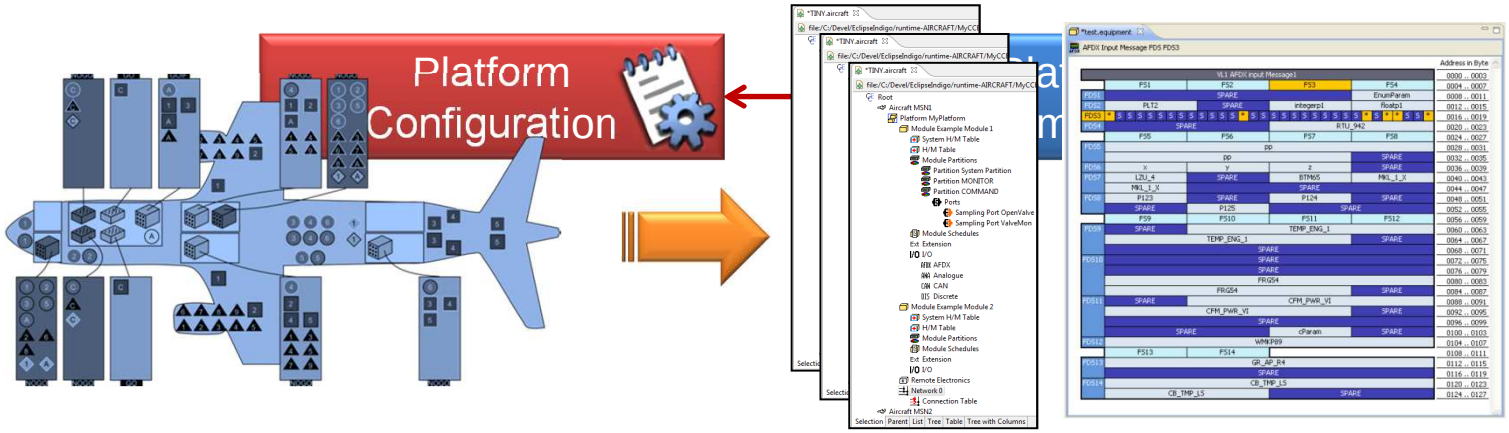
Transition between tools: Model2Model



Transition between tools: Model2Model

- Architecture Optimisation **results**
  - Generic module types
  - Abstract link types (data&power)
  - Simple routing (signal based)
  - Used very **early** in the process
- Configuration **needs**
  - Specific modules (CPM, RDC…)
  - Specific I/O channels (AFDX…)
  - Detailed routing/signal parameters
  - Used very **late** in the process

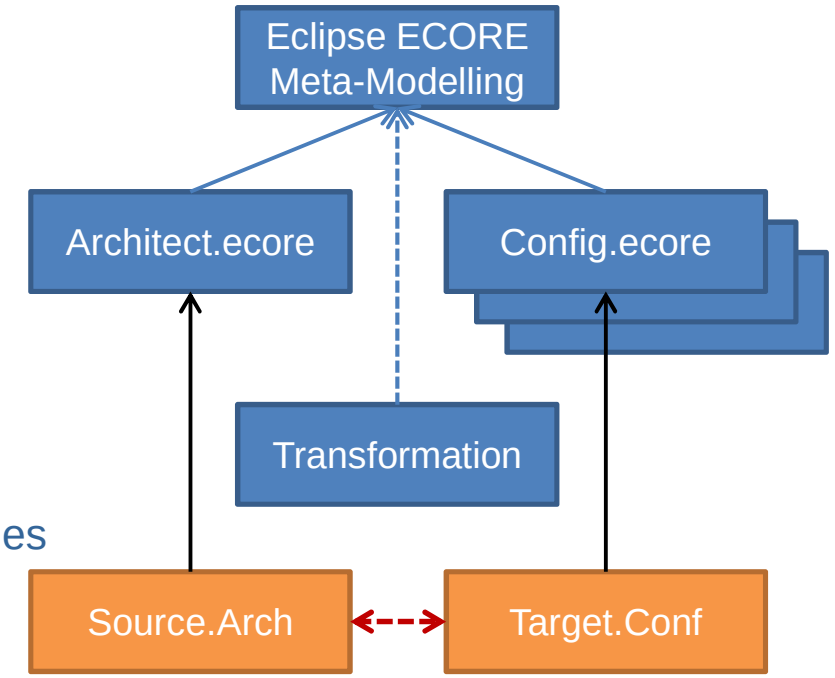
- **Auto-generate** IMA configuration stubs
- **Auto-retrieve** IMA architecture templates



- Methodology: **Model2Model transformation** between meta models
- Benefit: Error-free transition of information (re-use) between tools

Transition between tools: Model2Model

- Basis:
  - Both Frameworks are based on Eclipse
  - Both Frameworks share the same meta-modelling language (ECORE)
  - There exists a tooling that allows to define a transformation „script“ to transform parameters from one into another tool

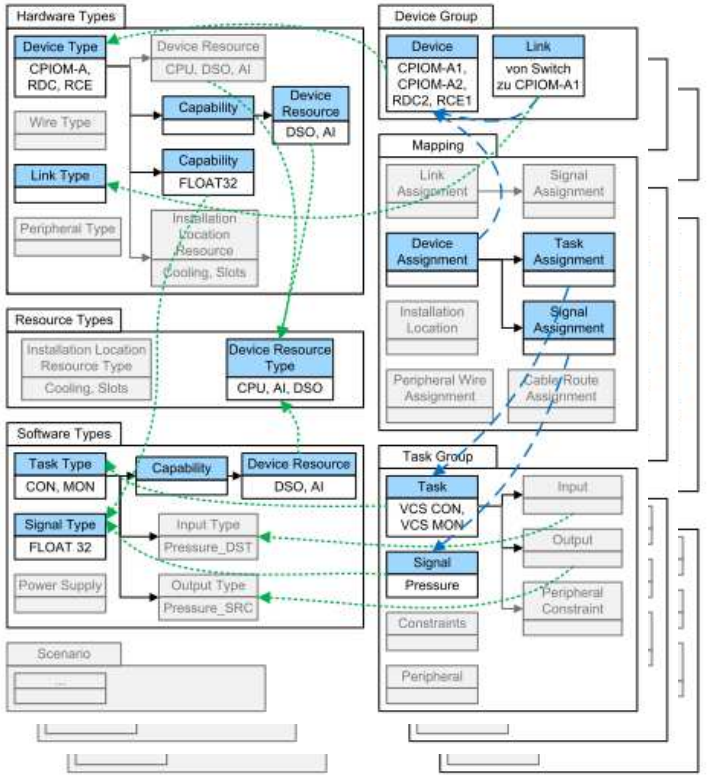


- Solution:
  - Different transformation technologies/languages
  - Some are file (XML) based, some are meta-model based

- Integration into Tools
  - Transformation script is code-generated and embedded as plugin into both tools
  - This results in the functionality

Transition between tools: Model2Model

- Result:
  - Investigated possible parameter transformations
  - Transform architecture → configuration stub
  - Transform configuration → architecture model
  - Developed sets of transformation scripts for both directions
  - Language with most benefits: “Query View Transformation operational” (QVTo)
  - By design, a 1:1 relation is not possible
    - ✓ Example for POT→PCT:
      In POT, device types are identified via string
      In PCT, devices are different classes
      → Solution: Naming rules
    - ✓ Example for PCT→POT
      No installation locations in PCT, but in POT (to compute cable length)
      → Data is missing after TraFo to POT
  - BUT: Better than starting from scratch!




```
//add default-child if TgtModule has no brotherObj
self.allSubobjectsOfType(SrcModule).oclAsType(SrcModule)->forEach(foundModule){
    var foundTargetModule := foundModule.resolveone(TgtModule);
    if (foundTargetModule <> null) then {
        if (foundTargetModule.brotherObj = null) then{
            foundTargetModule.childs += object TgtChildTypeA{
                name := "default";
            };
        } endif;
    } endif;
};
```

Any questions?

Conclusions

- Introduction of the configuration process
  - Complex, distributed process
  - Requires harmonisation to avoid iterations
- Introduction of Meta-Model and PCT
  - Supports several module types and actors
  - Modern software framework, based on plugins
- Transition between tools: Model2Model
  - Supports automatic, model-based transition of data between PCT and POT

Contacts

Martin Halle

Martin.Halle@tuhh.de

Nesspriel 5
21129 Hamburg
Germany

Tel.: +49 (0)40 42878 8209

Avionics Systems Hosted on
a distributed modular electronics Large scale dEmonstrator
for multiple tYpe of aircraft

Call identifier: FP7-AAT-2013-RTD-1

Project co-funded by the European Commission within the Seventh Framework Programme (2013-2017)