Model-based Transition of IMA Architecture into Configuration Data

This paper will introduce a model-based, formal transition of Integrated Modular Avionics (IMA) architecture models to configuration blueprints.
IMA has been introduced in aircrafts since years and has become a common standard. As more and more systems implement their functions on IMA new demands arise on IMA hard- and software as well as the overall development process. The challenges for industrial use include seamless tool chains used to design, configure and integrate systems on IMA. Research activities within the European ASHLEY project (Avionics Systems Hosted on a distributed modular electronics Large scale dEmonstrator for multiple tYpe of aircraft), besides others, focus on the harmonisation of the IMA development processes and tools. Selected results are presented hereby.
IMA provides a generic hardware platform comprising a standardised hardware/software interface and generic module management functionalities. During IMA architecture design hardware platforms (i.e. IMA modules) are chosen that fit best to the requirements of the system applications hosted on IMA. Furthermore, hardware and software mappings are designed that optimally assign IMA functions to the hardware. Architecture evaluation in combination with optimisation is a complex task. Recently, model-based approaches and tools have been developed that allow multi-criteria optimisation for complex assignment activities.
Depending on the chosen architecture and assignment configuration documents have to be developed. These define the binding of application software to IMA hardware modules by means of resource negotiations in the first place. After that, way more detailed for integration of the applications on the selected IMA platform. The detailed configuration of IMA modules is split into a global and a local part. The global part describes the partition envelope and the local part describes the details of the binding between the Operation System (OS) and the application, including the I/O specification. Configuration documents are required to generate the load (application + configuration) for IMA modules. Recently, a model-based configuration engineering framework has been introduced that allows for a less error-prone development process of configuration documents at aircraft level across several modules and system applications.
For configuration the information of architecture definitions is not consequently re-used in the present tool chain. Configuration documents are created (mostly table based) from scratch which is not efficient at all. It is time-consuming, prone to errors due to misinterpretation and hard to trace and to maintain.
To bridge a gap between early architecting and configuration engineering and owing to the fact that both frameworks presented are model-based, transition techniques have been investigated and developed. These implement an automated transfer of relevant architecture into configuration data and vice versa. This allows to consequently re-use architectural information to create configuration document blueprints and/or retrieve architecture models from existing configuration documents e.g. for the use of a re-assignment or in update programmes.
Within the first part of this publication the process of architecture definition and configuration engineering is analysed. The existing approaches and especially the model-based tools for these process steps are explained with special focus on their modelling- and parameter domain.

In the second part of this publication an approach is presented that makes use of modern model-based transformation techniques to interchange existing specifications between the selected tools. Both tools share the same Eclipse platform and are based on the Eclipse Modelling Framework (EMF). EMF allows defining parameter domains by means of meta-models, a well-known concept in software engineering. Especially for EMF, multiple transformation frameworks exist that allow defining transitions from one into another meta-model or parameter domain in general. The major advantage is a formalised, automated and thus error-free transition between both worlds. Furthermore, a well-defined transition specification can be easily maintained and adopted to future needs without the need of additional complex software development. There are, however, some drawbacks. Both domains (architecture and configuration) contain parameters that are not part of the respective other domain. For example, architecture models contain detailed installation location information not relevant for configuration and configuration includes a lot of specialised I/O parameters not relevant for the architecture. Making use of the most promising transition frameworks and techniques available this publication will explain in detail what information can be transformed, how and at what cost. The methods presented will be compared to a conventional industrial approach.

Author: Martin Halle (Martin.Halle@tuhh.de)