Title: Efficient modelling of avionics systems: combining standard language and custom editor

Authors: Raphaël Faudou (Samares Engineering), Patrick Farail and Laurent Duffau (Airbus), Camille Louge (AtoS),

Keywords: SysML, TOPCASED, Model based system engineering, Embedded system, Papyrus

Introduction

In aeronautic, space, and automotive fields, critical embedded systems are more and more subjected to high safety requirements and so need an important verification effort during the development phase to avoid errors. It becomes more difficult for industrial actors to specify such systems, while ensuring the quality and the non-ambiguity of the specification. Model Based System Engineering (MBSE), allows designers to mature specifications quicker than with document approach because errors and inconsistencies in requirements are generally detected earlier thanks to their formalization in model elements. MBSE approach is more and more recognized as a good approach in industry and we find a lot of evaluations in R&T programs.

But a good approach is not enough for wide dissemination in operations. Industry is now looking at tools to support MBSE efficiently: modelling cost should not be a too high cost in comparison with "traditional" document approach or the MBSE benefits will be annihilated by MBSE language and tool learning curve. Finally MBSE dissemination is directly linked to the ability to provide efficient modelling. In the rest of this document we describe the different strategies leaded by Airbus for two levels of Embedded system development activities concerning MBSE tooling: system operational scenarios and functional analysis, software detailed design. Then we present the last generation of tools developed, that combine Domain specific and standard modelling language for maximal modelling efficiency.

Different strategies on MBSE tooling at Airbus during last decade

At software level, the first approach from Airbus was to use standard modelling language (UML) to support MBSE for obvious reasons: benefit from diversity in modelling tools, benefit from a large panel of engineers already trained on those languages learnt at school/university, find a lot of training material (books) and consulting if needed and get a unified and unambiguous semantics.

Airbus introduced UML ten years ago (it was UML 1.3) on one avionic equipment for A380 aircraft program and chose a commercial modelling tool to support UML.

Usage of models was mainly motivated by improving communication and teams only used a few diagrams (Use Cases, Sequence, classes) to help them illustrating their specification and design documents. Requirement traceability was managed by two kinds of links:

- Links between classes specialized into "requirement" and use cases
- Links between classes specialized into "requirements" and classes specialized into "test case"

This approach helped reducing the textual part of specification and architecture documents but modelling remained tedious and sometimes inefficient and error prone. Behaviour was modelled with different levels of detail and with different concepts according to the person in charge of modelling. UML tool was too large with too many concepts that even with guidelines it was hard to ensure consistency between diagrams and for the whole model.

Thanks to the TOPCASED open source project based on Eclipse platform, Airbus learnt a lot on techniques and technologies to develop modelling editors and realized that they could specify and develop their own modelling editors at low cost. To support their home specification method (based on SA/RT) they decided to create a dedicated language or DSML (Domain Specific Modeling Language) called “SAM” to describe 'to describe the functional software decomposition, the associated flows (control, data and message) and behaviour (through automata)..

As this language was simple and as editor was fully generated on the basis of this language (Model Driven Architecture), the SAM modelling editor was simple to use (limited number of concepts) and there was good adoption at software level and even in other departments including Design office. Requirement traceability was not managed with this language and managed through another tool (TOPCASED Req) that was able to create links between two different models: requirement model resulting from requirement document import and SAM model.

But this approach was not perfect. SAM language and SAM tool were easy to develop but they faced two major issues:

- Lack of extensibility: contrary to UML that is extensible by nature through the “profile” mechanism, SAM had no way to extend language except use of “eAnnotations” what is a low typed technique to add elements. It remains possible to check those “extended” concepts but tooling is more complex and differs from other concepts. Impacts are everywhere (model check, document generation, code generation...). Another solution is to modify the SAM meta-model but this implies to regenerate a new version of the tool and to manage possible inconsistencies, then migration of legacy of models, what can become a very complex task according to the evolutions (concept divided into two new concepts, change in the multiplicity of a relation…).

- Not standard: as language was not standardized, it was difficult to get feedback to improve it and to find engineers that had learnt it. Same for the tool. Even if tool was open, Airbus could not share efforts on tool support and maintenance as they were the only users...Yet another Editor to maintain...

In parallel, there was a lot of progress on technology and frameworks to support UML customization, especially through Eclipse MDT papyrus component.

With Papyrus 0.8, released in July 2011 it became possible to customize the palette of diagrams at user level to hide some tools and to add other tools created by extending UML concepts (stereotype concept).

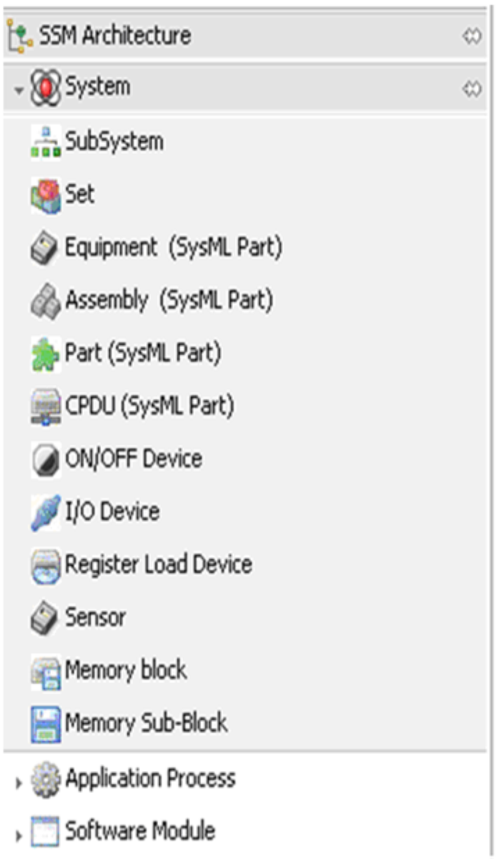
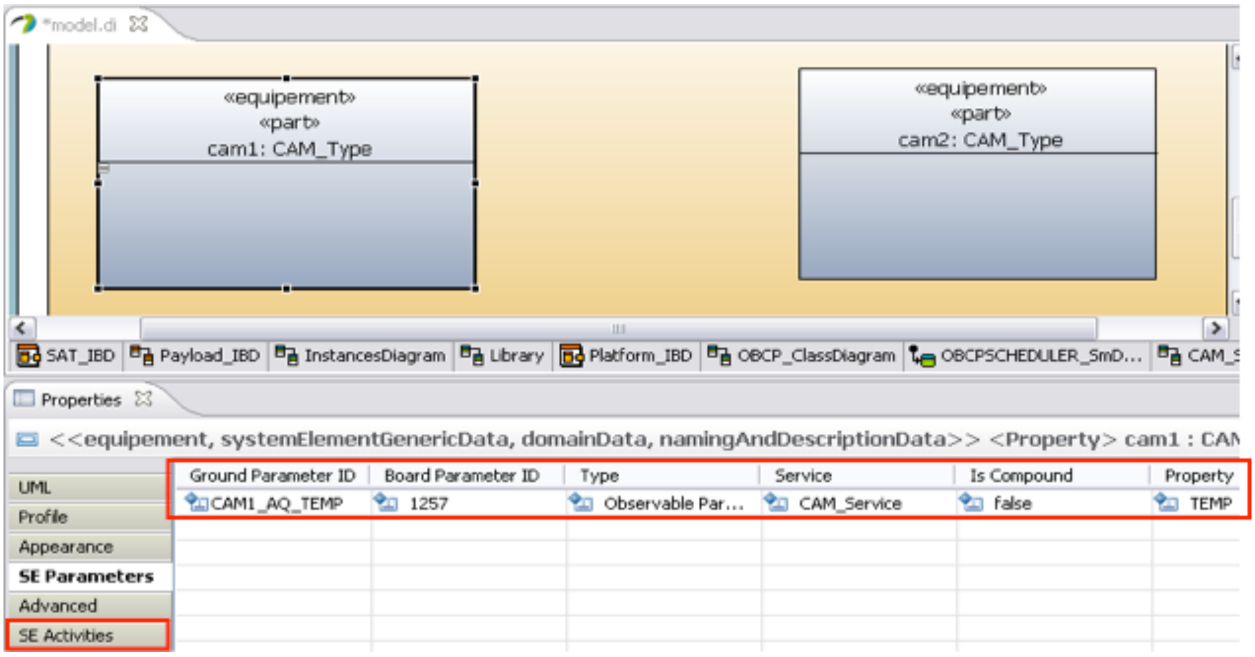


Figure 1: Example of palette customization through a profile designed for space industry

The property view that is contextual to the selection in a diagram could be customized at programming level but with low effort (simple programming interfaces to implement or to extend).



<<equipement, systemElementGenericData, domainData, namingAndDescriptionData>> <Property> cam1 : CAN

Ground Parameter ID	Board Parameter ID	Type	Service	Is Compound	Property
CAM1_AQ_TEMP	1257	Observable Par...	CAM_Service	false	TEMP

Figure 2: Example of customization of property view to provide easier navigation to parameters and activities of a System Element

And the model explorer view could be customized to hide some “technical” model elements or to change style of some key elements.

Two modelling experiences above standard editor

Two years ago Airbus EYYW department (Avionics Software) had opportunity to develop a new modelling tool to support design with “HOOD” method (this method has been deployed for 20 years to design embedded software with very good feedback). They decided to use UML language for standard conformance and all associated benefits, but with strong customization to provide “simple” modelling editors adapted to end user vocabulary and process, based on EMF (Eclipse Modeling Framework) and on existing editor to get benefit from the work done in the TOPCASED[1] projects. We present this tool called “TOPHOO” in a following paragraph.

In the Design Office, for modelling aircraft systems architectures, , the approach was a little bit different. 10 years ago there was no modelling standard to describe architecture and systems engineers mostly used Visio tools. Several R&T projects have proven the interest of a top/down functional approach, a first set of COTS tools has been deployed : CORE from Vitech and Rhapsody from IBM.. CORE was a good choice for readability and usability as the tool was really designed in order to support systems engineers. No long list of “technical” properties as in UML tools, no need to define the type of data when linking data to operations or functions. No long list of possible semantics on links as in UML.Rhapsody was the classic UML/SysML tool. The SysML language was defined in 2007 by OMG by extending UML (with a standaed profile) to support the representation of Systems concepts. Even if there are only 8 diagrams in lastest SysML (12 for UML 2.4), using SysML remains a complex activity for systems engineers as they have to understand the different concepts and choose which diagram to use to describe their architecture. And SysML provides generic concepts like “block” while systems engineers would like to manipulate “operations” and “functions” and “physical items”. In the end, and in the move to extended deployment of MBSE, the systems designers need easy to use tools that are using business instead of generic concepts.

The AGeSys French collaborative project was the opportunity to specify and test a new tool called “Scade System Designer”, based on SysML but with shortcuts for easier access by systems engineers not used to SysML language. This solution is promising and gives interoperability with SCADE suite software models that are used in Airbus to develop several critical systems. For the moment, the support of SysML language is limited to only structural view with Block dedinition and Block internal diagrams.In parallel, and in order to support the definition of a future generic Airbus MBSE methodology for system architecture, the choice was done to customize the existing Topcased/papyrus framework in order to adapt it to the system needs. This prototype, called “FAST” tool supports edition of operational and functional architecture models that were strongly expected by Airbus systems engineers in different R&T programs. The main goals were:

- A modelling tool as simple as possible but that remains SysML compliant,
- Develop a generic prototype that supports Airbus methodology definition and especially edition of operational models and functional models (systems functions) with data flow and system breakdown.

[1] http://www.topcased.org

FAST tool is presented later in this document.

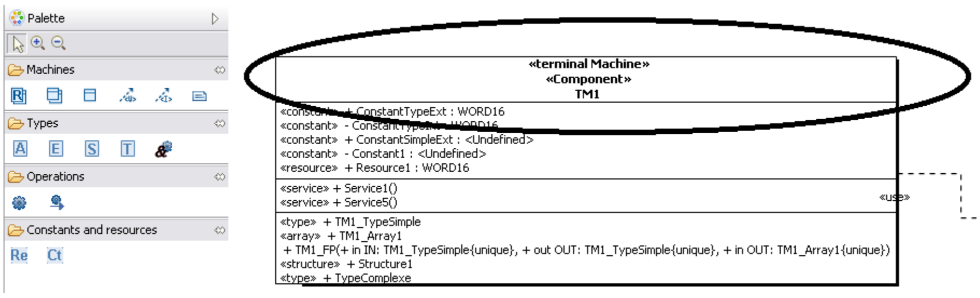
As for software level, choice was done to use Eclipse MDT Papyrus for SysML and customization through profile mechanism as key drivers of the solution.

TOPHOO an Airbus UML tool customized to support HOOD methodology at software level

TOPHOO provides a simple way to define component breakdown structure to prepare the coding phase. Components are called "Machines" in Airbus terminology. There is one root machine (the equipment or subset to address), and several sub machines, and terminal machines (at the lowest level). Each machine can define different elements:

- Services (operations) with some of them that are exported (made public)..
- Types
- Constants and resources

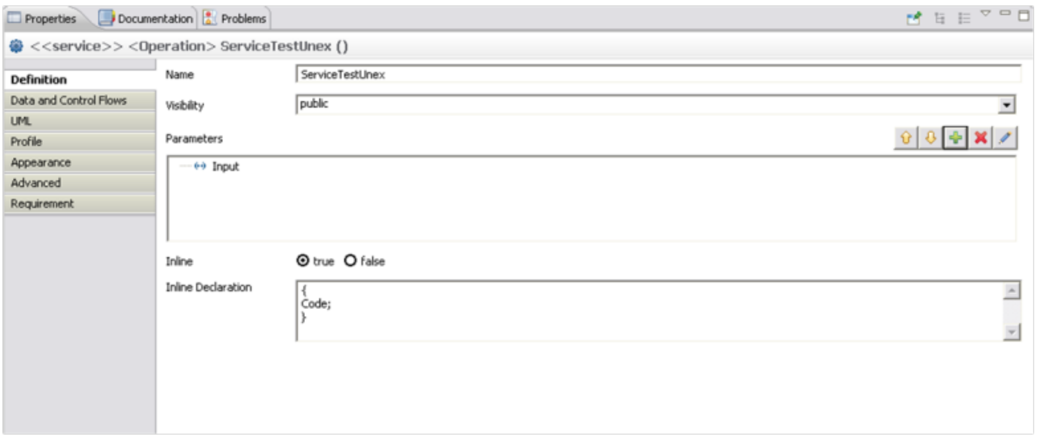
As you can see, the palette presents those elements. Behind the scene there are UML elements but they are completely customized through the usage of dedicated UML profile so that vocabulary displayed is HOOD for end user (not UML).



```
«terminal Machine»
«Component»
TM1
«constant» + ConstantTypeExt : WORD16
«constant» - ConstantTypeInt : WORD16
«constant» + ConstantSimpleExt : <Undefined>
«constant» - Constant1 : <Undefined>
«resource» + Resource1 : WORD16
«service» + Service1()
«service» + Service5()
«type» + TM1_TypeSimple
«array» + TM1_Array1
 + TM1_FP(+ in IN: TM1_TypeSimple{unique}, + out OUT: TM1_TypeSimple{unique}, + in OUT: TM1_Array1{unique})
«structure» + Structure1
«type» + TypeComplexe
```

Figure 3: HOOD concepts available in TOPHOO (diagram palette and nodes)

Property view is contextual and is adapted to highlight important parameters in first two tabs. UML properties are hidden or available only through advanced tab.



<<service>> <Operation> ServiceTestUnex ()

Name	ServiceTestUnex
Visibility	public
Parameters	Input
Inline	[✓ true] false
Inline Declaration	{ Code; }

Figure 4: TOPHOO customized property view : service definition

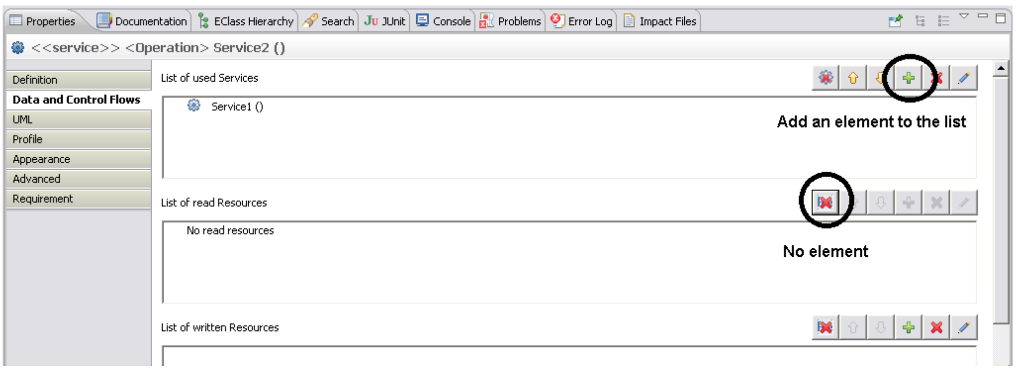


Figure 5: TOPHOO customized property view: data and control flow

From TOPHOO it is possible to manage requirement traceability, generate documentation, check consistency rules, generate C code .

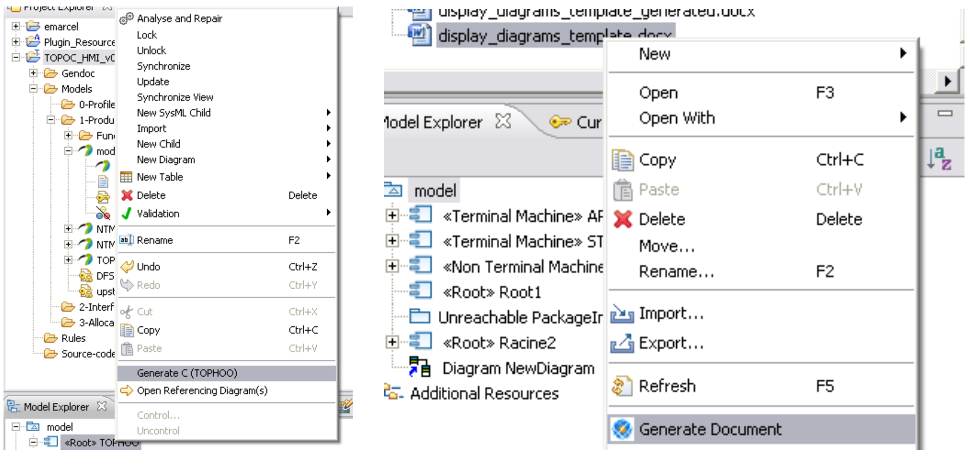


Figure 6: C code generation and document generation from TOPHOO model

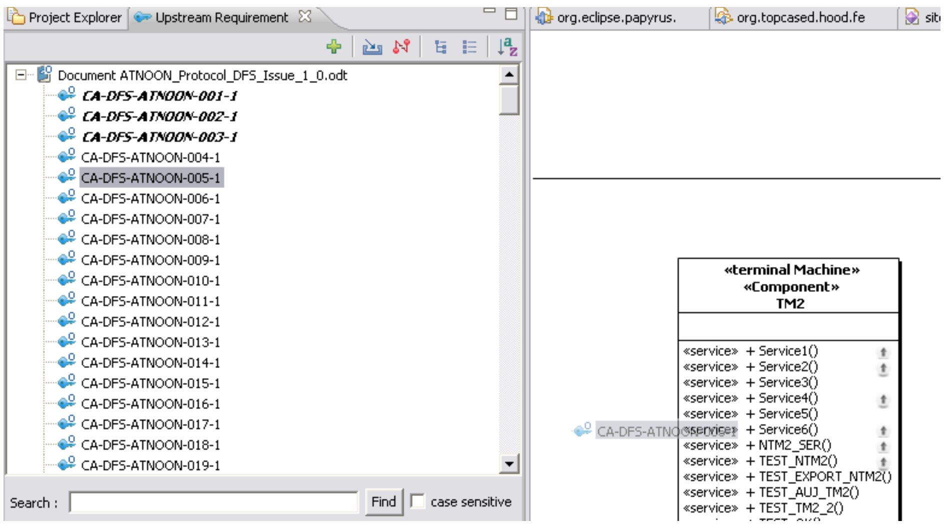


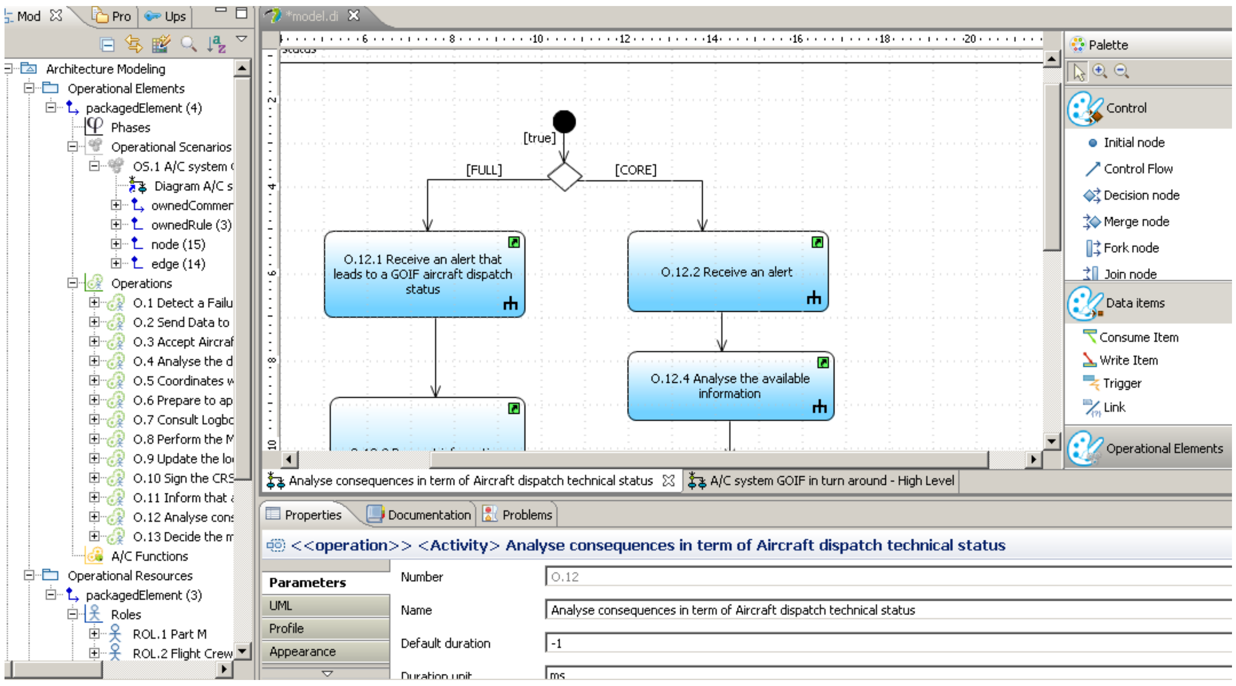
Figure 7: Illustration of traceability link created by "drag and drop" between a requirement and a service

TOPHOO solution has been industrialized and is used in production on a software equipment of A350 aircraft program (safety level = C).

FAST, an Airbus experiment of customizing SysML to support architecture operational & functional views at system level

FAST has two different views: operations and functions.

Operational view provides way to create operational scenarios, manage operation breakdown structure and reference data called items as inputs, triggers or outputs. It is possible to define the sequence of operations, decisions, fork/join... and reference functions.



<<operation>> <Activity> Analyse consequences in term of Aircraft dispatch technical status

Number: O.12
Name: Analyse consequences in term of Aircraft dispatch technical status
Default duration: -1
Duration unit: ms

Figure 8: operational view in FAST

Functional view provides a way to decompose system function into sub functions and to manage the way data/energy is produced and propagated through the different functions.

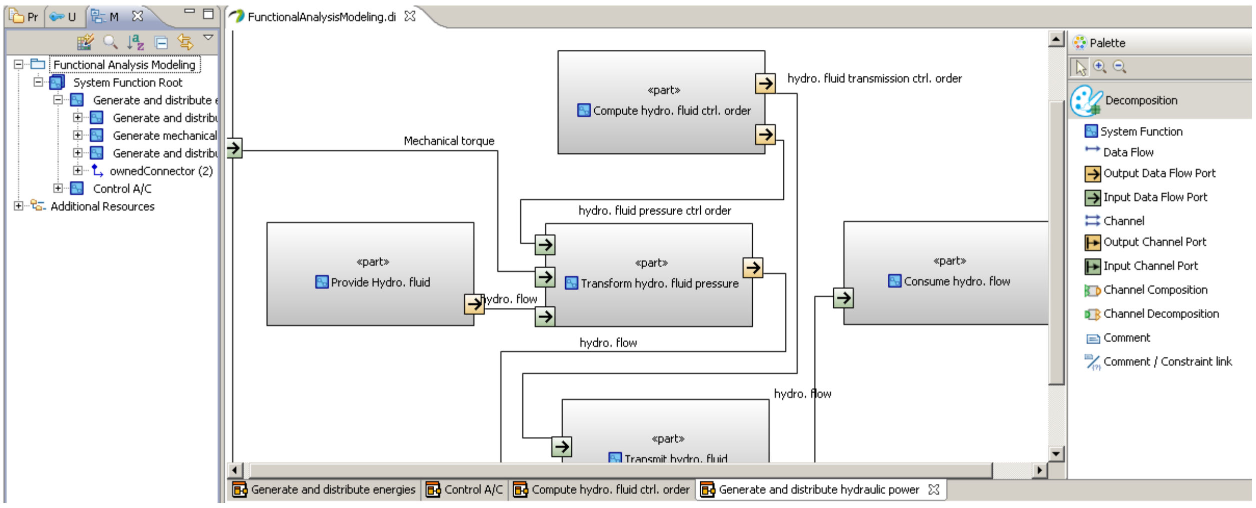

FunctionalAnalysisModeling.di
Functional Analysis Modeling
System Function Root
Generate and distribute
Generate mechanical
ownedConnector (2)
Control A/C
Additional Resources
«part» Compute hydro. fluid ctrl. order
hydro. fluid transmission ctrl. order
Mechanical torque
hydro. fluid pressure ctrl order
«part» Provide Hydro. fluid
«part» Transform hydro. fluid pressure
«part» Consume hydro. flow
hydro. flow
«part» Transmit hydro. fluid
Palette
Decomposition
System Function
Data Flow
Output Data Flow Port
Input Data Flow Port
Channel
Output Channel Port
Input Channel Port
Channel Composition
Channel Decomposition
Comment
Comment / Constraint link
Generate and distribute energies
Control A/C
Compute hydro. fluid ctrl. order
Generate and distribute hydraulic power

Figure 9: functional view in FAST

FAST is currently used by system designers on R&T programs, with good operational feedback.