Model Driven Approach for the Ariane 5 launcher

ASTRIUM Space Transportation

Philippe Gast
Journée SysML France - Toulouse
27/11/09
Overview

- Introduction
- SysML/UML modelling
- Model validation
- Multi-user / traceability management
- Conclusion
Astrium: part of EADS, a global leader in aerospace and defence
Astrium’s activities are based in three key areas

**Astrium Space Transportation**
The European prime contractor for civil and military space transportation and manned space activities

**Astrium Satellites**
A world leader in the design and manufacture of satellite systems

**Astrium Services**
At the forefront of satellite services in the secure communications, Earth observation and navigation fields
An impressive product and capability portfolio

- Launchers: Ariane, Soyuz, Rockot, Vega
- Ballistic missiles, missile defence
- Future launchers
- Orbital systems: Columbus, ATV, Operations, Atmospheric re-entry systems
- Propulsion & equipment
- System design, system integration & production
Introduction

- **Context**
  - Study co-funded by CNES
  - For the Ariane 5 program

- **Objectives**
  - Formalization of system requirements allocated to the software
  - Early system requirements validation
  - Increase of the documentation consistency
  - Facilitate the code development

⇒ MDA: Model Driven Architecture
Steps

- **Targeted systems**
  - Representative functionalities of the Ariane 5 program
  - Focus on vehicle and mission management
    (GNC part covers by another study)

- **Choice of a tool**
  - RSM / RSA: not mature enough at the study start
  - Tau G2: not adapted for embedded systems
    (according to the tool provider)
  - **Rhapsody: Selected**

- **System modelling & validation**
- **Code generation**
- **Process**
• Acquisition of measurement
• Where am I?
• Where shall I go?
• Compute the commands
• Send commands to actuators

Event driven (spacecraft management)

Data flow driven (control / command algorithms)

Matlab/Simulink
Scade

SysML/UML Rhapsody
Overview

- Introduction
- **SysML/UML modelling**
- Model validation
- Multi-user / traceability management
- Conclusion
Documentation tree

System Design (mission management)
- Textual documents
- Redundant information (copy & paste)
- Inconsistency
- Ambiguity
- Incompleteness

System Design (GNC)

Software specification
Capture of the system requirements allocated to the software

HOOD modelling

Software design
Document to be reviewed by system engineers

Manual coding

Coding

Document to be reviewed by stage engineers
The UML model becomes the unique reference
All the documentation is generated from the UML model
General Approach

The System Engineering process in SysML shall be split up in three main phases:

- **First level functions identification**: analysis of the stakeholder requirements, main functions definition and translation into system requirements (functional and quality of service requirements when possible at this level). The system boundary shall be clearly identified and the external actors too. The traceability links between system requirements and upward requirements could be partial and completed in the next two steps.

- **Second level functions identification**: each previous main function is split up in several second level functions. Requirements of the first level are affected to the second level functions after decomposing and refining process if needed. Scenarios (nominal and contingency) shall be written to detail the function behaviour. Only the needs of external interfaces shall be captured at this step (black box view).

- **System design**: for each previous second level function, the inputs/outputs interactions shall be exhaustively described (implementation of the external view needs specified during the previous step). The internal view (white box view) shall also be detailed in this phase.

The steps 1 & 2 are mainly under system team responsibility whereas the step “System design is realized in co-engineering by system and software team.”
MODEL ORGANIZATION: A5 example
Use Case diagram: first level
MISSION STATES:

**GROUND**
- Reactions:
  - \( ev_{\text{Ground}} \) (ignite VULCAN 2)

**EAP_FLIGHT**
- Reactions:
  - \( ev_{\text{H0}} \) (ignite EAP)

**GDP_FLIGHT**
- Reactions:
  - \( b = H0 + \text{delta_tdeo} \) (ignite EAP)

**EPC_FLIGHT**
- Reactions:
  - \( ev_{\text{K1}} \) (ignites Release)
  - \( ev_{\text{H2}} \) (stop VULCAN 2)

**ESC_FLIGHT**
- Reactions:
  - \( ev_{\text{H2}} \) (ignites HM 780)

**BAL1_FLIGHT**
- Reactions:
  - \( ev_{\text{H1}} \) (payload Release)

**BALFIN_FLIGHT**
- Reactions:
  - \( ev_{\text{H4}} \) (payload Release)

**Typical Key events:**
- \( H0 = \text{time} \)
- \( H0 + \text{delta_tdeo} = 7.05c \)
- \( H1 = \text{low threshold level detection} \)
- \( K1 = \text{guidance event} \)
- \( K2 = \text{guidance end} \)
- \( K2\_1 = \text{mission event} \)
- \( H2 = \text{payload release} \)
- \( \text{mission duration} = 1H \)

**An ECA launcher can only perform a type 1 mission. (Type 2 is forbidden for ECA launcher)**

**Two mission types are possible:**
- **type 1:** no intermediate ballistic phase between EPC separation and the first ECS ignition
- **type 2:** intermediate ballistic phase between EPC separation and the first ECS ignition.
2nd level: sequence diagram
2nd level: internal block diagram
SysML / UML modelling

- Only the following SysML elements and diagrams are allowed inside a second level function package.
- No decomposed Use Cases are authorized at this level because a second level function is considered as the terminal level of decomposition (Use Case diagram is forbidden).

<table>
<thead>
<tr>
<th>Elements</th>
<th>Diagrams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case</td>
<td>1st level only</td>
</tr>
<tr>
<td>Part</td>
<td>Internal block diagram</td>
</tr>
<tr>
<td>Requirement</td>
<td>Requirement diagram</td>
</tr>
<tr>
<td>Part / Use Case</td>
<td>Sequence diagrams</td>
</tr>
<tr>
<td>None (self standing diagram)</td>
<td>Activity diagram</td>
</tr>
</tbody>
</table>
System modelling status

- **SysML model developed**
  - All the information entered in the model
  - All documentation automatically extracted from the model (use of “ReporterPlus”)

- **Advantages**
  - Same language use between teams
  - Easily updated documentation
  - Consistency between documents
  - Documentation reviewed by
    - System vehicle engineers
    - Stage engineers

- **On going**
  - UML model to be developed by system engineers
Overview

- Introduction
- SysML/UML modelling
  - Model validation
- Multi-user / traceability management
- Conclusion
Model Validation

 Tentative to use statechart to model the dynamic behaviour
   Not enough readable

 Creation of a generic model
   Sequence simulation and time constraints
   Easy use
   Executable model

 Generic model use to simulate each modelled functionality
   Fast implementation
   Link to specifications
   Possibility to launch project simulator
Modelling of a time window mechanism with statechart

Not intuitive reading
Model Validation

- Definition of treatment using generic model

The generic model provide high level features, such as time window mechanism.
Model Validation

- Simulator animation
  - Generic model for the system under development
  - Animated Statecharts (for the environment)
  - Comparison between expected and generated sequence diagrams

- Test scenarios creation
  \[\Rightarrow\] specification errors detection preceding conception phase
Overview

- Introduction
- SysML/UML modelling
- Model validation
  - Multi-user / traceability management
- Conclusion
Multi-user management

- Model organized in several sub-projects

- Configuration management
  - Interface between Rhapsody and ClearCase
Traceability Management

- Rhapsody Gateway / Reqtify use
  - Connecting documents (Word, Doors, Rhapsody…) together
  - Import / export of requirements

- Generation of the traceability matrix
Overview

- Introduction
- SysML/UML modelling
- Model validation
- Multi-user / traceability management

Conclusion
Conclusion

- Capture of system requirements dedicated to the software
  - UML/SysML models shared by system & software engineers
  - Reviewed and approved by stage engineers

- Early validation of system requirements
  - Generic model
  - Environment model
  - Simulation

- Refinement toward the software
  - Automatic generation of the software architecture
  - Dynamic code can be used only for simulation