Model based approach in configuration data management for LVCUGEN Flight Software

S. Duprat[1], A. Haugommard[1], J. Galizzi[2], C. Navarro[3], M. Masmano[3]

(1) Atos, (2) CNES, (3) Fentiss

Introduction

Configuration data are a key element in generic software frameworks, and more especially in TSP based software architecture. This is the case with LVCUGEN, a CNES generic Flight Software solution composed by a hypervisor and several partitions. Each component is highly configurable, making this solution generic and facilitating adaptation for a mission specific final product. We will present how the model-based approach has been chosen and implemented to guarantee the respect of the qualification perimeter of each component as well as the consistencies between all configuration data of the software products in the context of ECSS level B standards. The approach covers model verification and code generation from model and integrates a qualification strategy. The methodological solution is adapted to a collaborative organization where the product is developed and integrated by increments.

Context

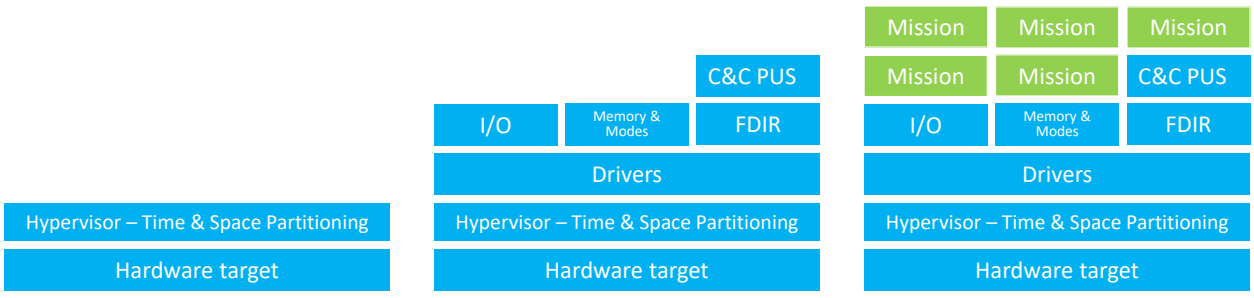
The technical context is those of Flight Software with the CNES solution LVCUGEN. This solution is a product line of Flight Software based on a component based approach used in the context of Integrated Modular Avionics (IMA), using Time and Space Partitioning (TSP) architectures. In this approach, the Flight Software is composed by different partitions orchestrated by a hypervisor ensuring deterministic scheduling and inter-partition communication. For the LVCUGEN solution, the hypervisor is Xtratum Next Generation (XNG), provided by the Fentiss company.

LVCUGEN provides already developed partitions for I/O, memory management, FDIR, Command & Control following PUS standard and drivers covering different hardware platforms (LEON, Zynq). This solution has been developed by different companies for CNES and is now maintained by Atos.

XNG is a hypervisor developed by FentISS that provides services needed by safety-critical systems such as partition management, support for system and non-system partitions, resource virtualization, temporal partitioning based on a cyclic scheduling policy, spatial partitioning based on the support provided by the hardware, inter-partition communication through sampling and queuing ARINC-653 like ports. The configuration of the system is provided as a set of XML files with a series of restrictions. The configuration files not only allows the user to determine the structure of the system (memory areas, partitions, devices…), but also allows to configure the observability and the health monitoring service, which is in charge of detecting faults in the hardware or in XNG. On the top of XNG, FentISS also provides LithOS, a guest operating system with an ARINC-653 APEX, XRE, a C library, and the BSP for RTEMS and Linux, an interface between the operating system and XNG.

The solution is also a host platform where standard partition can be completed with mission specific missions.

Lifecycle of the development of a LVCUGEN Flight Software is incremental and involves different actors. This lifecycle is meaningful for configuration data aspects.



First layer corresponds to the Hypervisor Layer. This layer is already highly configurable and configured	Basic software and other standard partitions are coming on top of the Hypervisor. All of them are configured and shall be consistent between each other and with the hypervisor configuration.	Standard partitions are completed with mission specific partitions which, in their turn, can have a configuration that shall be consistent with lower layers.

Incremental steps can be repeated each time a dedicated partition coming from diverse contributors is integrated. During every steps, the integrator is in charge of checking that all software parts are well integrated, that each software component is configured in an authorized configuration i.e. each software component is used in compliance with its qualified usage domain.

Needs & Technical solution

The situation depicted by the context and the lifecycle highlights the importance of the configuration data, the respect of the qualified configuration perimeter of each component (Usage Domain) and the guarantee of consistencies across the different configurations. The technical solution is a line product, with a high degree of configuration and lots of configuration elements. These elements are not centralized but distributed in software components in a TSP architecture.

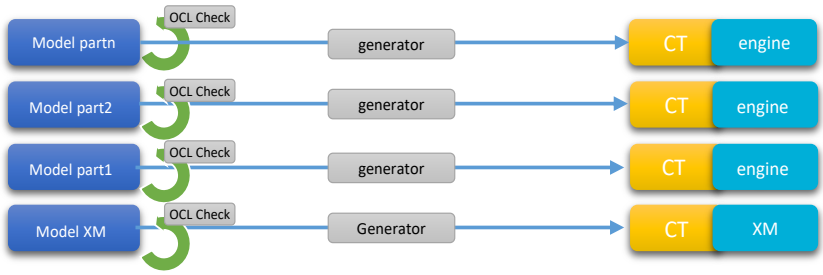
In this context, a model-based solution is a good candidate to cover these needs, avoiding a long and tedious manual verification of all the configuration parameters at each evolution. Based on a previous study, the technical solution is under deployment for the LVCUGEN context.

The solution consists in relying on the formal expression of models to represent the configuration with a common and standard structure facilitating collaboration and co-engineering, while benefiting from constraints expressions to check consistencies on these configuration elements. The checks are performed through the expression of constraints relying on the powerful and formal OMG Standard OCL language (Object Constraint Language). With a simple example, the checking of the XML model on the right with a unicity constraint expressed very synthetically with OCL through an invariant would result in a violation of the rule *Parameter_uniqueKey* for element E2.

```
-- In any configuration Element , two Parameters cannot have the same tuple (id1,id2)
context Element
inv Parameter_uniqueKey : self.parameters -> isUnique(id1+'_'+id2)
```

```
<configuration>
 <element name="E1" >
  <parameter name="param2" id1="1"  id2="A"/>
  <parameter name="param4" id1="2"  id2="B"/>
 </element>

 <element name="E2" >
  <parameter name="param1" id1="3"  id2="A"/>
  <parameter name="param2" id1="3"  id2="A"/>
 </element>
</configuration>
```



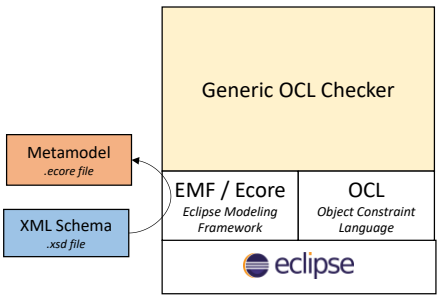
The approach relies on an integrated chain including code generation to bring configuration defined into models up to the source code level. Qualification concerns have to be handled for both verification and generation features. We will make a focus on these aspects below in the presentation.

The model engineering chain is fully composed on well-proven standards: initial XML models are handled as Eclipse EMF models, checked through OCL Constraints, and code is generated through Acceleo model-to-text standard based on OMG MOFM2T language.

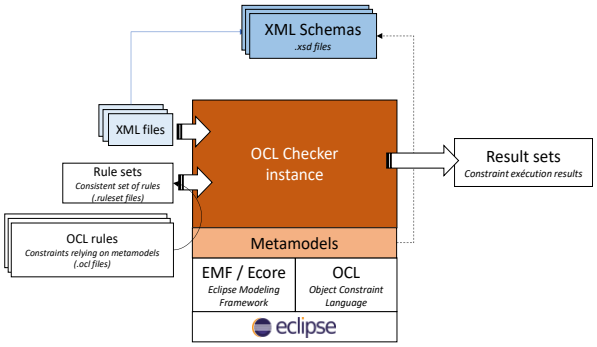
Detailed application

Once main principles are presented, the interest of the solution lies in some details that we must present, particularly for metamodel verification and incremental constraint definitions and verifications.

The verification feature is defined at two levels. This has consequences for both users and the qualification.



First level of the OCLChecker tool corresponds to the generic features to automate OCL checks, manage OCL rules by rulesets, produce result through the Eclipse GUI and also in batch mode. This first level is generic and is not linked to any OCL rule nor domain model.



This generic tool is instantiated by each software provider with its dedicated meta-model(s) and OCL rules, enabling domain-oriented evaluation of constraints and consistency checks.

For LVCUGEN configuration, final users can benefit from a domain-oriented checker integrating on top of the generic OCLChecker all necessary meta-models for the particular software of interest and can enrich an initial set of consistency and check rules specific to their context.

Qualification issues

Qualification issues have been studied in the scope of ECSS-Q-ST-80C for the European Space engineering and space standards and also ED-215 / DO330C coming from aeronautics domain and offering qualification strategies for multidomain engineering tools.

Two features shall be addressed for the qualification point of view: qualification of the checker tool and qualification of the generator.

(1) Qualification strategy for the checker tool

The Tool Qualification Level is defined depending on the level of the product software and on the features of the software (can it introduce a bug or just fail to detect a bug). Conclusions is TQL 5 for this tool. The verification strategy is important. It has been chosen a two layers qualification strategy, in accordance with the architecture of the tool. We applied the concepts defined in ED-215C of Developer TOR (Tool Operation Requirements) and User TOR as follows:

1/ The generic part of the tool is considered as a COTS and generic features are defined in a development TOR. A set of tests is associated for verification purpose.

2/ Each application on a domain with a metamodel and some OCL rules leads to a User TOR and other elements of a qualification kit. Each user shall replay all verifications tests for both levels, in its operational context.

We can notice that in this strategy, the tool is qualified with its OCL rules, which make a very big difference compared with a strategy of qualifying the full Eclipse EMF OCL stack.

(2) Qualification strategy for the code generator

The code generator is clearly identified as a GAC in the development process of the product. In this case, the double chain approach has been chosen. The double chain shall be applied with other technologies.

Feedbacks

Implementation of this technical solution on the LVCUGEN project including the hypervisor part demonstrated that the OCL language is well adapted to the verification of configuration data. All expected constraints have been well implemented.

It appeared that usage of metamodels was enough in some cases to implement constraints that can be defined by structural properties. In this case, verification is obtained by a consistency check between the model and the meta-model. Other cases are addressed with specific OCL rules.

One of the major outcomes of the tool is the ability to work on several models that can rely on different metamodels.

This has been implemented with:

- a model at the Hypervisor level with its own schema defined by an XSD file

- a model at the partition level, relying on its own metamodel
- OCL constraint verifying consistencies between both models (example: a partition is using a port whose name is declared in the hypervisor configuration).

All features presented are integrated in the development toolchain, with Continuous integration based on Gitlab and Jenkins (depending on the actors).

Lastly, the main feedback comes from the collaborative aspects, as the solution has been designed for this purpose. Collaborative aspects have been demonstrated with the contribution of 3 organizations.

Once the technical solution was available, Fentiss has applied it for its own hypervisor configuration model. For that, Fentiss has instantiated the generic tool for its own metamodel and has implemented all required OCL rules for the configuration restrictions.

Before the use of this tool, FentISS verified the configuration with a non-qualified tool, which made necessary a manual verification of the hypervisor configuration. With the use of a formal language for the verification, FentISS can provide an automatic verification of the hypervisor configuration according to the OCL rules.

Additionally, the use of this tool offers a new feature to the user. He can add his own rules along the provided system rules. That is, the user could check that his hypervisor configuration matches both the system restrictions and his own created restrictions in order to have a correct configuration of his defined system.

FentISS provides the elements required for using the tool as a part of the hypervisor product. The elements provided are the rules in OCL formal language at the annex of the hypervisor user manual, the rules in a OCL file, the hypervisor system schema in a XSD file and the metamodel of the hypervisor for the tool. Regarding future developments, we are strongly interested in exploring the use of XNG as target execution platform integrated in the TASTE framework. TASTE is an open source Model-Based Development framework sponsored by ESA which supports formal models and automatic code generation. It provides an open source toolchain for embedded software development. Using XNG as TASTE target execution platform could improve the safety of mixed-criticality systems based on the IMA approach.

Atos, commissioned by CNES for the LVCUGEN developments, defined the metamodel for the LVCUGEN partitions, OCL rules at partition levels and OCL rules for consistency checks between partitions and hypervisor.

CNES integrated each partition, configured the whole software and verified that each constraint is satisfied, including also project specific constraints. Prior to that, the tools (checker and generator) were built by Atos for CNES.

Dissemination, other applications

The OCL Conf Checker tool is available under https://github.com/CNES/OCL_Conf_Checker.

Missionization of product lines

For each instantiation of LVCUGEN, the integrator will be in charge of setting up the global configuration of the complete mission software, composed by several generic components (XNG, IOS, MMDL, HSEM, CC, …), and mission specific partitions/applications. This global configuration will have to satisfy the following rules:

- Respect of the Usage Domain of each component
- Consistencies across configurations (mainly valid connections of the components between each other)
- Mission specific configurations to guarantee the memory, CPU budgets and guarantee the qualification perimeters of each component, in the frame of this specific mission.

This last item also eases the emergence of product lines, where, for example, the configuration perimeter of a star tracker partition can be maintained (and guaranteed) across different missions, despite evolutions of its neighbors' partitions.

References

- EMF (Eclipse Modeling Framework): https://www.eclipse.org/modeling/emf/
- OMG OCL: https://www.omg.org/spec/OCL
- Acceleo: https://www.eclipse.org/acceleo/
- OMG MOFM2T: https://www.omg.org/spec/MOFM2T/1.0
- OCL Conf Checker tool: https://github.com/CNES/OCL_Conf_Checker