Towards a Model-Based Approach to Support Physical Test Process of Aircraft Hydraulic Systems

Ouissem Mesli-Kesraoui, Yassine Ouhammou, Olga Goubali, Pascal Berruet, Patrick Girard, Emmanuel Grolleau

Towards a model-based approach to support physical test process of aircraft hydraulic systems

MESLI-KESRAOUI Ouissem[1,2,3], OUHAMMOU Yassine[2], GOUBALI Olga[1], BERRUET Pascal[3], GIRARD Patrick[2], and GROLLEAU Emmanuel[2]

[1] SEGULA Engineering, France
[2] LIAS/ISAE-ENSMA and University of Poitiers Chasseneuil du Poitou, France
[3] Lab-STICC/University of Bretagne Sud LORIENT, France
ouissem.mesli@ensma.fr, yassine.ouhammou@ensma.fr, olga.goubali@segula.fr, pascal.berruet@univ-ubs.fr, patrick.girard@univ-poitiers.fr, grolleau@ensma.fr

Abstract. The physical integration of an aircraft consists of the assembly of several complex subsystems (including hydraulic systems) developed by different stakeholders. The cleanliness of the developed hydraulic subsystems is ensured by performing several decontamination and flushing tests. This testing phase is very tedious as it is mainly performed by SCADA (Supervisory Control and Data Acquisition) systems and depends on chemical substances. However, as the design is mainly expressed in informal textual languages and synoptic diagrams, this testing is currently done manually and is determined by the experience of the testers. This makes it error-prone and time-consuming.
In this paper, we propose to capitalize the effort for physical testing of hydraulic systems by proposing a model-based system engineering approach that allows: (i) to graphically specify the systems under test and (ii) to automatically generate the corresponding test cases. A proof of concept is proposed as well as a case study.

Keywords: DSML · MBT · Avionic Test · Hydraulic System · Flushing

1 Context and Positioning

The aircraft is composed of complex subsystems called avionics systems such as: the hydraulic power system (ATA 29), the landing gear system (ATA 32), the flight control system (ATA 27), etc. [4]. A hydraulic system is a complex assembly of hydraulic, electronic, and mechanical components. This system uses a pressurized fluid to transfer energy from one point to another [6]. Before commissioning an avionics system, a series of tests should be performed to ensure that it is free of contaminants. The presence of contaminants in a hydraulic circuit can alter its function and damage it. To reduce this damage, the hydraulic system and its components must be flushed [6] to remove contaminants that may have been created during the components fabrication, assembly and/or maintenance.

Flushing is one of the best and most cost-effective solutions for removing contaminants from hydraulic circuits. For efficient flushing, the system is decompressed in loops (a portion of the circuit consisting of equipment and piping) [8].

To perform the flushing test, the testers follow three steps (Fig. 1). In the first step (Specification Translation), the testers obtain and translate the specification documentation needed to identify the test loop. These documents are essentially: (i) The physical architecture of the system under test. It is a Piping and Instrumentation Diagram (P&ID) which illustrate the connections between different the equipment and piping; (ii) The flushing process and constraints. The flushing process consists of a description of the components and their roles: the component to be flushed, the generating components, the component that stores the fluid, valves to circulate the fluid, etc. A set of constraints to be respected like the velocity, loop order, etc. are also described in constraints documents. In the second step (Identification of abstract loops), testers use the above documents to divide the circuit into sections or loops (part of the circuits on the physical architecture). This ensures that the velocity is maintained (test constraints compliance). Finally, in the third step (Translation of abstract loops to concrete loops) the testers convert the abstract loops into executable loops by adding information about the role of each component and how the components are handled (opening the generating component, operating the valves to allow the fluid to flow towards the loop, etc.). These concrete loops are then executed on the SCADA test bench according to a specific sequence to ensure that the fluid only flows through cleaning sections.

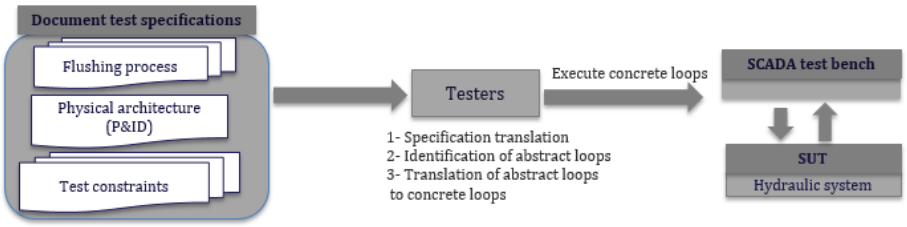

Fig. 1. Testing of the avionics hydraulic system

The manual definition of test loops after the above steps is fraught with several problems. First, the translation process (step 1) is a tedious task, since each piece of information (physical architecture, flushing process, and constraints) must be translated into a different language. In fact, nowadays, the information about the test process and test constraints is mainly scattered in a document-based manner (e.g., text and Excel files with natural languages), and the information about the hydraulic system is schematic-based (e.g., Visio and PDF files) without any domain semantics. Second, abstract and concrete loop identification are often written manually by testers. Writing tests manually is a tedious, time and resource consuming task and often prone to omissions and errors.

The problem of identifying flushing loops and their location has attracted much attention in the context of drinking water decontamination [8, 7]. However, the solutions proposed for drinking water decontamination cannot be applied to hydraulic circuit flushing. Firstly, hydraulic circuits are complex and, unlike water circuits, have many different components, and secondly, the test conditions of hydraulic circuits are very different from those of water circuits.

To adapt to the increasing complexity of avionics system testing and to reduce the manual effort required for testing, MBT was developed to make it easier for testers to test avionics systems and to focus their efforts on the SUT and the features under test [10]. MBT enables the automatic generation of test cases from system models. Otherwise, while the MBT method has been used with success in automating certain tests of avionics systems [1, 10], it has never been used in generating flushing loops.

In this paper, we propose a model-based approach that leads to correct-by-construction test cases (flushing loops) to reduce tester effort. This can shorten the time-to-market as designers and testers can work together on data-centric models. For this purpose, we first propose a Domain Specific Modeling Language (DSML) to play the role of a pivot language to design and collect all the information of the hydraulic system under test, as well as the flushing process and constraints. The DSML is intended to improve the reusability of hydraulic system test bench designs and to strengthen semantics. Second, we use the paradigm Model-Based Testing (MBT) to automate the generation of test cases (loops) from instances of the proposed pivot language.

The rest of the paper is as follows. In section 2, we introduce our DSML and show its use for the automatic generation of flushing loops for hydraulic systems. In section 3, we evaluate our approach using a case study and discuss the obtained results. Section 4 concludes this paper and discusses some future research directions.

2 Our contribution: A Reusable & Optimized approach for flushing avionics systems

Due to the lack of space, in this paper we focus on two contributions corresponding to the first two steps of the flushing process. The first one consists in defining a new pivot language to express the P&ID of the hydraulic system, the flushing process and the flushing constraints in one language. The second contribution is the automatic generation of the abstract loops for the flushing test.

2.1 Towards a pivot language

A DSML is a language tailored to the needs of a particular domain. It allows to create models of complex systems that are closer to reality and use domain specific vocabularies to facilitate the understanding of different designers [9]. The use of domain concepts and terms allows experts to understand, validate, modify, or even create a DSML.

The abstract syntax of our pivot language is modelled by a metamodel augmented by several OCL (Object Constraint Language [3]) constraints that capture domain rules.

Core-elements of the proposed abstract syntax. Our pivot language (Fig. 2) allows modeling the physical architecture of a hydraulic system, the functions that ensure the flushing process, and the test constraints.

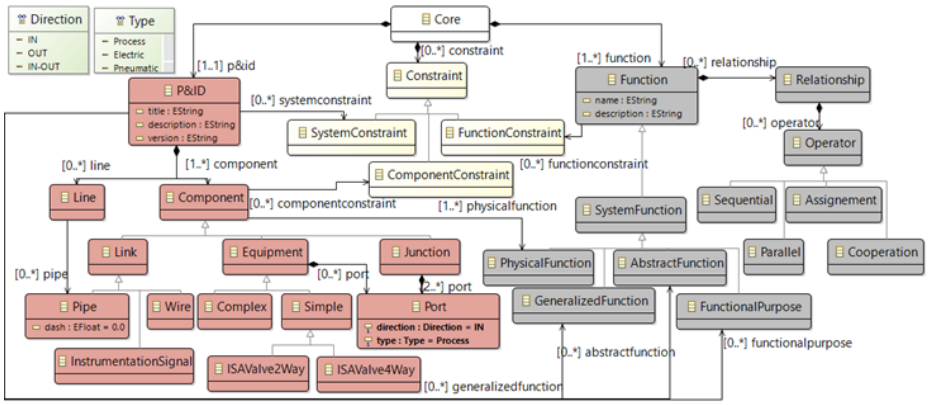

Fig. 2. Core elements of the proposed abstract syntax

The physical architecture facet. The root class in the physical architecture metamodel is `P&ID` (left part in Fig. 2), identified by a title, description, and version. The P&ID consists of a set of components that may be `Equipment`, `junction` or `link`. An equipment can be `Simple` or `Complex`, which consists of other simple equipment. A junction can connect equipment by their ports (In, Out, etc.) to a link. A `Pipe` link is identified by a diameter (dash).

We also added several OCL constraints to capture domain rules. For example, the following constraint limits the number of process ports to 2 for components of type `ISAValve2ways`.

```
1  Context ISAValve2Way inv:
2  self.port->count(MM::Type::Process) = 2
```

The function (test process) facet. Each `Function` (right part in Fig. 2) is identified by a name and a description. `SystemFunction` are functions performed by the system. Functions are divided into (`FunctionnalPurpose, AbstractFunction, GeneralizedFunction and PhysicalFunction`). The first one describes the high level functions of the flushing process, this function is always the purpose of the design of the test bench (e.g. cleaning the circuits) which is

adopted by the whole SUT. The `AbstractFunction` is used to describe the function that is actually to be realized by the system (flushing). The `Genralized-fucntion` is adopted by a subsystem of the SUT (e.g. Isolate the loop). And the low-level function is the `PhysicalFunction` inherent to the elementary components (Supply hydraulic power). Functions are connected by operators that allow the execution order of functions. There are several operators: `sequential` (functions are executed sequentially) and `parallel` (functions are executed in parallel). To assign a `PhysicalFunction` to a component, the `Assignment` operator is used. A `GeneralizedFunction` is guaranteed by more than one component, for which the `Cooperation` operator is provided.

The constraint facet. A `Constraint` may refer to a function, a system, or a component (Fig. 2). For example, the `FunctionalConstraint` describes constraints that refer to system functions.

Testers designing test model. We develop a graphically concrete syntax to our pivot language that allows information to be represented in a familiar form to testers. In the first phase, testers used the proposed tool to design P&ID of the hydraulic system under test by dragging and dropping components and connections stored in a proposed library. During the creation of the P&ID, testers must specify properties for the different components (diameter, length, supported pressure, etc.). Once the P&ID is created, the testers can specify the different functions that describe the flushing process and the flushing constraints.

2.2 Graph and abstract loop generation

We propose an MBT-based approach to identify the different flushing loops in the hydraulic system (Fig. 3). This approach consists of 3 steps. In the first step, the modelling of the P&ID, the flushing process and the test conditions is performed by our proposed tool (graphical concrete syntax). The resulting instance is called *test model*. The second step consists in transforming the test model into an oriented graph. For this purpose, each P&ID component (valve, cylinder, pump, etc.) is translated into a node and the piping into edges between nodes. In our case, loop identification remains to identify a path in the generated graph. To identify the loops, the components of P&ID are divided into: *source*, *well*, *waypoint*, and *goal*. The ***source*** component is the component that supplies hydraulic energy to the loop. The ***goal*** component is the component that needs to be flushed. The component ***waypoint*** is a component through which the fluid must necessarily pass, and the path ends in the component ***well*** that recovers the flushed fluid. The generated graph is used in the third step to derive the abstract loops of the flush. For this purpose, a search algorithm (Dijkstra algorithm [5]) is used to find a path from the source component to the target component and from the target component to the well component. Specifically, this step calculates the different loops needed to flush the hydraulic system.

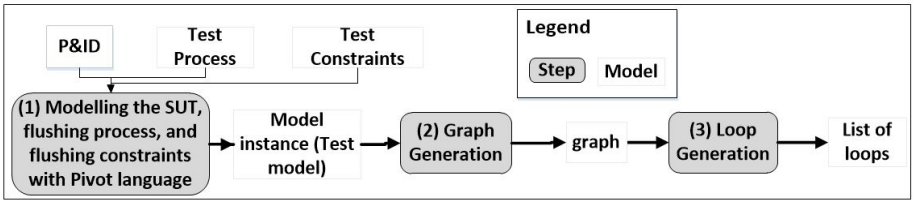

Fig. 3. MBT-based flushing loops generation

3 Proof of concept

To emphasise our proposal and show its importance, we use the landing gear avionics system [2]. Fig. 4 shows the P&ID of the hydraulic circuit that controls the landing gear and associated doors [2]. It contains 3 sets of landing gear (front, right, left). Each set consisting of a door, the landing gear and the hydraulic cylinders. This system is powered by the general Electro-valve which is connected to four Electro-valves. The first valve controls the closing of the different doors (front, right, left) and the second is used to open these doors. The third and fourth valves are used to extend and retract the three landing gears (front, right, left). The opening/closing of the doors and the retraction/extension of the landing gear are done by cylinders.

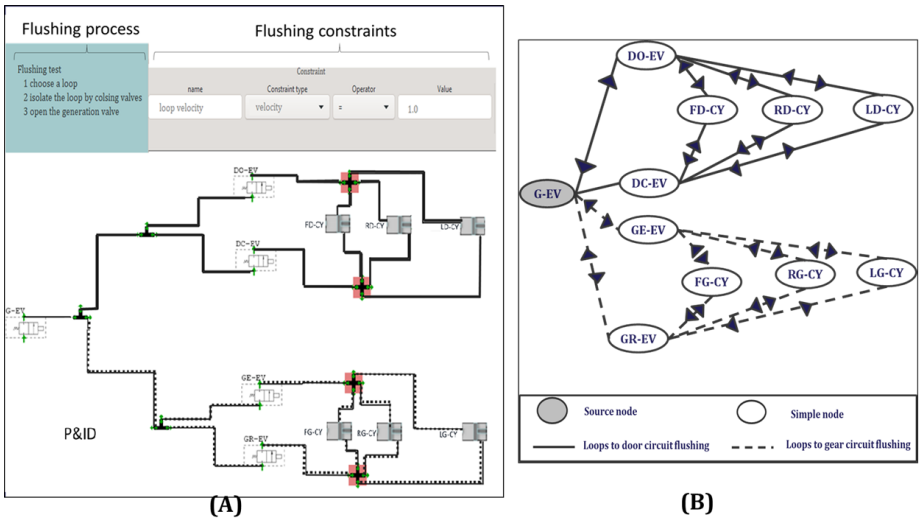

Fig. 4. (A): The modelling of the three facets of the Landing gear system with our tool. (B): Graph generated from the test model

3.1 Method

On this system, we applied our MBT-based approach to generate the flushing loops. First, we used the proposed modelling tool to capture all the required information about the flushing of the hydraulic system. We design P&ID in the new pivot language using the component library included in the modelling tool (Fig. 4). Then we specify the flushing process (functions), and the flushing constraints to obtain the Model Instance (test model). After the modelling step, we generate an oriented graph from the pivot language specification. The obtained graph of the landing gear system is shown in Fig. 4. The various steps of Dijikstra's algorithm are applied to this graph. The purpose of flushing in this system is to flush the different cylinders (LD-CY, RD-CY, DF-CY, LG-CY, RG-CY and FG-CY) corresponding to left-door cylinder, right-door cylinder, front-door cylinder, left-gear cylinder, right-gear cylinder and front-gear cylinder respectively. These cylinders correspond to the goal component. The loop used to flush a target component consists of two paths. A pressure path starts at the source component and ends at the goal component. A return path starts from the goal component to the well component.

3.2 Results & Discussion

The pivot language proved to be very useful as it successfully separated what the tester was doing (specifying) and how the test was implemented (generating loops). For the landing gear system example, our approach generates 4 loops. These loops are represented by full and dashed lines in Fig. 4. Each loop is chosen to be isolated to be flushed separately. Two full line loops allow the door circuit (front door cylinder, right door cylinder, left door cylinder, and various piping between these components) to be flushed by alternately operating the closed and open Electro-valve. In fact, if the close Electro-valve is used to circulate fluid from the General Electro-valve to supply the front door cylinder, right door cylinder, left door cylinder and various piping components, then the open Electro-valve is used to return fluid from the cylinders to the General Electro-valve, and vice versa. the second loop takes the opposite path (Electro-valve is used for supply and the close Electro-valve is used for return). The dotted line loops (two) are used to flush the front gear cylinder, the left gear cylinder and the right gear cylinder by selecting the supply valve and the return valve (selecting between the extend Electro-valve and the retract Electro-valve).

We then compared these generated loops with the manual loops created by testers and they are correct. This result proves that our pivot language contains all the necessary information to generate all the required loops for the hydraulic system. Nevertheless, it will be useful to improve the calculation process. For this purpose, some constraints must be considered in the calculation process: (i) consider the flushing constraints (velocity, flow, and pressure conditions) on the optimization algorithm; (ii) add a success constraints that is used to evaluate the cleanliness of a flushed component. If a loop is unsuccessful, that loop must be re-executed.

4 Conclusion

Testing is the most important step that cannot be neglected in the design of aircraft hydraulic systems. This step is always preceded by the design of test loops. The latter can be time consuming and tedious, depending on a large number of components and constraints that need to be considered. Our goal is to reduce this design effort by providing an approach that allows testers to easily define test models (P&ID, test constraints, test process) and generate the required loops. We first proposed a pivot language to unify the specification of all facets in one language. This language is supported by a proposed tool that is closer to the domain. Using this tool, experts from different domains can easily create specification models without learning a new language. Thus, the work of experts is focused on the specification. To generate loops from specifications, we proposed an MBT-based approach to determine the constitution of a loop.

5 Acknowledgement

We would like to thank Mr. Mikaël LE ROUX expert from SEGULA Montoir De Bretagne-France, for his collaboration and his involvement.

References

1. Arefin, S.S.: Model-based testing of safety-critical avionics systems (2017)
2. Boniol, F., WIELS, V., Aït-Ameur, Y., Schewe, K.D.: The landing gear case study: challenges and experiments. International Journal on Software Tools for Technology Transfer **19**(2), 133–140 (Jul 2016). https://doi.org/10.1007/s10009-016-0431-4, https://hal.archives-ouvertes.fr/hal-01851720
3. Cabot, J., Gogolla, M.: Object constraint language (ocl): a definitive guide. In: International school on formal methods for the design of computer, communication and software systems. pp. 58–90. Springer (2012)
4. Collinson, R.P.: Introduction to avionics, vol. 11. Springer Science & Business Media (2012)
5. Dijkstra, E.W., et al.: A note on two problems in connexion with graphs. Numerische mathematik **1**(1), 269–271 (1959)
6. ISO 16431: Hydraulic fluid power — system clean-up procedures and verification of cleanliness of assembled systems. Standard ISO 16431:2012(E), International Organization for Standardization (2012)
7. Poulin, A.: Élaboration de procédures d'intervention en réponse aux contaminations se produisant en réseaux d'eau potable. Ph.D. thesis (2008)
8. Poulin, A., Mailhot, A., Periche, N., Delorme, L., Villeneuve, J.P.: Planning unidirectional flushing operations as a response to drinking water distribution system contamination. Journal of Water Resources Planning and Management **136**(6), 647–657 (2010)
9. Sloane, A.M.: Post-design domain-specific language embedding: A case study in the software engineering domain. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences. pp. 3647–3655. IEEE (2002)
10. Yang, S., Liu, B., Wang, S., Lu, M.: Model-based robustness testing for avionics-embedded software. Chinese Journal of Aeronautics **26**(3), 730–740 (2013)