Binding Fault Logic to System Design: a SysML approach

Kester Clegg

Department of Computer Science, University of York, U.K. E-mail: kester.clegg@york.ac.uk / rolls-royce.com

David Stamp

Rolls-Royce Controls PLC, U.K. E-mail: david.stamp@rolls-royce.com

John McDermid

Department of Computer Science, University of York, U.K. E-mail: john.mcdermid@york.ac.uk

To gain the benefits of MBSA within MBSE, traditional fault logic models need to align closely to the system's functional decomposition. To achieve this in a consistent fashion, we have developed a new SysML safety profile based on the functional blocks provided to us by system design. Functional deviations derived from the FHA (which is modeled in SysML using the same profile) are propagated between blocks through ports defined by the system engineers, and the associated fault logic (including hardware generated base events) is modeled in standard internal block definition diagrams. Functional failures represent only part of the failure model (which will always include events and scenarios outside the functional requirements), but it ensures that for functional deviations at least, fault logic models follow the functional hierarchy and use the same design blocks as the system engineers. It means that specification changes to functional blocks or ports can be picked up and flagged to the safety team as requiring inspection, and it enables direct traceability within the SysML repository of derived safety requirements from the PSSA / FHA through to the fault logic used to demonstrate acceptable mitigation of risk. We demonstrate the new profile's use in the context of a gas turbine control system design, and discuss the advantages and shortfalls it provides in an industrial setting.

Keywords: SysML, fault logic, fault trees, safety analysis, MBSE, MBSA.

1. Context

Traditional safety critical systems development often has a gap between system design (carried out as a requirements-based, functional decomposition) and safety analysis (concerned with modeling hazards, hazard mitigation, fault propagation, functional redundancy and other non-functional risk). Despite these different viewpoints, modeling the failure modes of the system should ultimately map back to the same understanding of the functional specification used by the system engineers. It is this common understanding that lay behind early claims that model-based systems engineering (MBSE) could act as a single repository of design documents that could meet the needs of both systems and safety engineers and reduce the amount of duplicated or out-of-date documentation used during development (Micouin (2014)).

Unfortunately support for safety analysis in system description languages such as SysML often lagged behind the primary concerns of functional specification and requirements capture (Biggs et al. (2018)). While it is true that via bespoke profiles SysML can be extended to model almost anything, that does not mean all the development processes will be able to fit coherently into a single data repository. What we mean by this is that being able to profile what you want to model is not the same as being able to model what you want from data defined and structured for another purpose.

2. Background

Model Based Systems Engineering (MBSE) has become the dominant paradigm in safety critical systems development with claims that it brings different modelling viewpoints and tool chains under the umbrella of a single model repository that forms the basis of all development and analytical effort. MBSE isn't tied to a particular system or architecture description language, but it has always targeted the needs of systems development (Lisagor et al. (2011)). The references listed here focus mostly on our own domain of safety critical civil aerospace development. However, even within this restricted sector there are several approaches based around particular languages: AltaRica (Boiteau et al. (2006), David et al. (2009), Rauzy and Blériot-Fabre (2014)), SCADE/Lustre (Joshi and Heimdahl (2005)), or around a modeling environment such as Matlab

Proceedings of the 31st European Safety and Reliability Conference
Edited by Bruno Castanier, Marko Cepin, David Bigaud, and Christophe Berenguer
Published by Research Publishing, Singapore.
ISBN: 978-981-18-2016-8; doi:10.3850/978-981-18-2016-8_718-cd

Simulink (Shao, N. et al. (2017)) in combination with other tools, such as HiP-Hops (Sorokos et al. (2015)) or physical simulation environments such as Modelica or Simscape (Schallert (2017), Shao, N. et al. (2017)).

SysML is an extension of the Unified Modeling Language (UML) that supports the specification, analysis, design, verification and validation of a broad range of systems. The intention is that a well-defined specification can fully describe the system, so that development and analysis can be performed using tools that take their data from a single source. Existing tool chains can be used provided that one can export data from the repository. Unfortunately, while a graphical interface for system modeling is supported by most tool vendors, a similar environment for safety analysts to model fault logic is rarely provided. Fault logic is often modeled using fault trees that trace faults from hardware failures (base events) to a top level hazard (see industry standard IEC 61025:2006).

In 2017 the OMG requested suggestions on how to represent fault trees in SysML as part of the Safety and Reliability Analysis Profile for UML, which will extend the SysML language with "the capability to model safety information" (Biggs et al. (2018)). As part of this, an early profile for Fault Tree Analysis (FTA) and Failure Mode and Effects Analysis (FMEA) has been developed and published (Biggs et al. (2018)) and is likely to form part of SysML 2.0. However, while the new profile is welcome it won't fit every company's internal processes and modeling needs. For example we have specific requirements at Rolls-Royce to model engine dispatch availability (the ability for an engine to carry a fault for a period of time). Given the previous lack of SysML support for fault logic modeling, it is not unexpected that some companies have gone ahead and developed safety profiles to fit their needs (see our previous reports: Clegg et al. (2019a), Clegg et al. (2019b)). The difficulty of trying to embed safety information in a SysML functional model is also discussed in Tanaka et al. (2020), with whom we have had correspondence over some of the issues.

3. Paper structure

Our experience suggests that being able to reproduce *existing* fault logic models in SysML is of little benefit in terms of improving the quality of analysis or reducing its development cost. For us to achieve those goals the fault logic models themselves had to fit more closely with the functional decomposition created by the system engineers, but even this can present difficulties. In order to illustrate this we provide a brief real world example in Section 4.1 of the step-by-step process of matching a fault tree structure to a corresponding functional decomposition. It should be noted these difficulties from the redundancy failure modes of a dual channel control system, as we describe in section 4.3, not from any difficulty in developing a bespoke profile that can capture fault logic in SysML. Having outlined the problem, we suggest an approach which restricts the fault logic model held in SysML to that which can be tightly aligned to the system's functional decomposition (Section 5). We give a brief overview of how the profile enables better traceability from the Functional Hazard Analysis (FHA) down to functional deviations caused by hardware component failures or events. Finally we discuss some of benefits and shortcomings of the approach with respect to quality and cost effectiveness (Section 6) and set out conclusions in Section 7.

4. The traditional viewpoint

A typical fault propagation model may build up a representation of failure logic based around functional redundancy, loss of function or the failure of some form of safety protection. In our case with a dual channel FADEC (Full Authority Digitial Engine Control), this means viewing the system as suffering loss of function on the channel in control. Loss of function (and functional deviation) is not often part of a functional specification except in a few specialized cases where protection is designed as a functional feature.

Traditionally it was sometimes claimed that having a separate failure model produced by the safety engineers was an independent check that bolstered the safety of the system. In MBSE, that view has shifted in favor of gaining quality and productivity advantages of the system development team working from a single source of design specification. To date, little evidence has been produced that demonstrates the MBSE advantages in this regard due to the issues of comparing like-for-like projects over long time spans. The lack of evidence with regard to process improvements and cost reduction is not something we can address in this paper, however our view is that quantifiable process improvements as a result of shifting to MBSE are hard to achieve over the lifetime of a single project.

4.1. *Adapting existing fault logic models to fit within MBSE*

Our previous work (Clegg et al. (2019b) modeling traditional fault trees in SysML and exporting the fault logic for analysis could be described as a success. However, trying to integrate the 'fragments' of fault logic with functional specifications (in the form of activity diagrams) was something we were not able to resolve satisfactorily. In this section we work through an example of this process, with specific reference to the SysML modeling environment at Rolls-Royce Controls. Some of the issues were tool specific, but most are down to

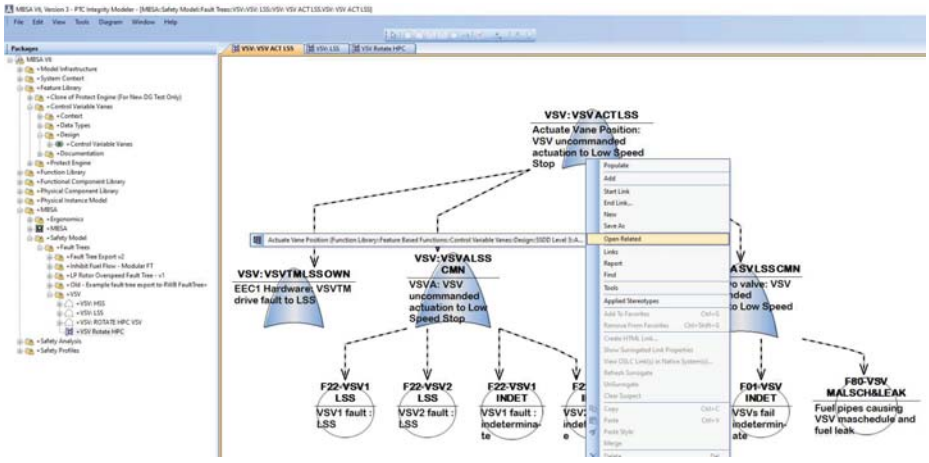

Fig. 1. Screenshot of our Fault Tree Profile user interface in PTC Windchill Modeler (formerly known as PTC Integrity Modeler, which is the version shown here) showing linking a 'fault tree fragment' to an associated diagram (functional specification activity diagram).

structural differences between the failure model and the system's functional decomposition. Our first example is taken from the UltraFan's Variable Stator Vanes (VSV) control and actuation system. The second example (Section 5) is taken from the ECOSIStem (Enabling Capability Of Software Intensive Systems) project on a smaller engine, with a slightly different SysML modeling standard as explained in the text.

The variable stator vanes control the air flow going into the turbine compression chamber, and their response is critical in the case of events such main shaft failure (shaft overspeed). There isn't space here to explain the detailed operation of the vanes, but the existing failure model was typical in that loss of function was modeled as loss of redundancy (i.e. the loss of the VSV function on both control channels). The functional specification for the VSVs is only concerned with how that function is implemented on either channel, the channel switching control logic is handled elsewhere. Our existing VSV fault trees had a lot of additional failure logic based on functional redundancy that was difficult to associate with the functional specification. In order to gain a better alignment, it became clear we would need to come up with a new 'channel neutral' failure model of the VSVs that focused on the failure modes of the function itself and removed issues of functional redundancy (so they could be modeled elsewhere).

4.2. *Matching fault tree fragments with activity diagrams*

The rationale behind attempting a channel neutral version of the VSV feature is to see whether the fault logic can be modeled so that fragments of the fault tree can be usefully associated with the corresponding part of the functional specification. In UltraFan this would be an activity diagram, while in ECOSIStem it would be an internal block diagram. Although it is possible to do this within the modeling tool (PTC Windchill Modeller) using a linked association (see Fig. 1, where the user right clicks on the fault tree fragment to access 'Open related' menu option). This is a weak association, in effect no more than a hyperlink. It does not have a formal connection in the model defined by a profile that could be used to check things like model consistency or verification of failure modes for system components.

But even with this weak association, trying to create a coherent model where fault logic is displayed alongside the corresponding functionality was difficult. The fault tree profile used a bespoke diagram type in order to display the fault logic's graphical format. This made it difficult to match 'levels' of functional hierarchy of the activity diagrams with fragments of a fault tree (these would correspond to 'pages' in an analytical tool like Reliabilty WorkBench's FaultTree+). But navigation also proved an issue. Fault logic fragments sometimes straddled separate activity diagrams, or a single activity diagram might require several fault tree 'pages' to be associated with it. This ended up being confusing for users. To try and illustrate this without going into detailed functional explanations, we walk through a brief example of the fault propogation path 'jumping' across a functional boundary as a user descends the functional hierarchy.

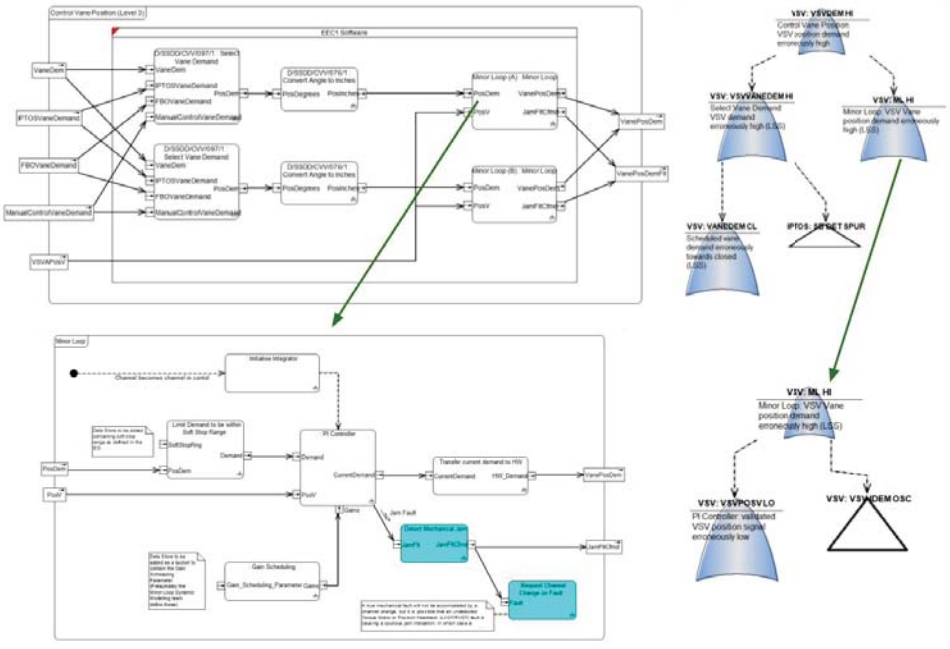

Fig. 2. Decomposition of the functional specification for Control Vane Position activity diagram (left) and its corresponding fault logic represented as fault tree fragments (right)

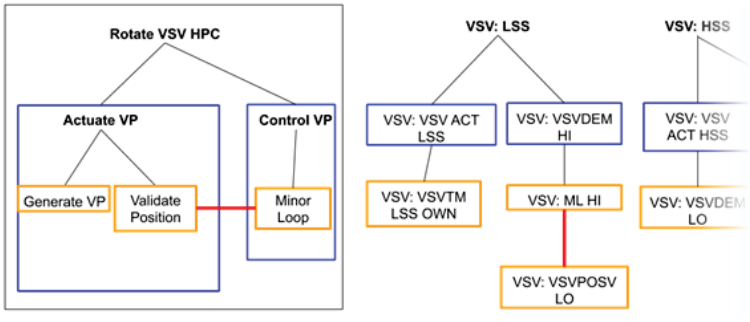

Fig. 3. Simplified view of the functional decomposition (LHS) and fault logic (RHS) in Fig 2. The red line shows how fault propagation jumps across the vertically defined functional hierarchy (shown by the blue boundaries).

4.3. *Decomposing the VSV function*

The Rotate High Pressure Chamber (HPC) VSVs control function is composed of Control Vane Position and Actuate Vane Position. Looking at the fault logic associated with the former we can start to deconstruct the fault logic model so that the fragments correspond to the functional decomposition (Fig. 2).

In this case, both the activity diagrams speci-

fying the functional decomposition and the Minor Loop is the lowest decomposition that the system engineers will go to (after this it is passed onto software or physical implementation). However, the fault logic needs to trace the failure as far as the base events generated by the hardware. Initially, this means it needs to trace why the vane demand was set erroneously high, due to the validated VSV position having been given as erroneously low, i.e. the failure stems from the vane position being not where the system claims it is. But which part of the functional decomposition does this failure mode relate to?

If we trace the fault logic through the functional hierarchy, the matching function below is not from any internal blocks to Control Vane Position. The function that traces that fault from VSV:POSV LO (vane position low) is associated with Validate Actuator Position, which is a component of Actuate Vane Position and ultimately part of Rotate HPC VSV.

To see how the lower levels of the VSV: Low Speed Stop (LSS) fault logic cuts across functional boundaries, we can draw a simplified tree showing hierarchical levels in the functional decomposition on the left and the associated branches of fault logic (as fault tree fragments - only the top gate name is shown) on the right (Fig. 3). The abbreviations refer to the slightly fuller names given in Fig. 2. The colors show our attempt to match the levels, and while not all the sub fault tree fragments can be shown, the linkage in red shows how descending the failure logic to the base event jumps across a functional boundary between Actuate and Control Vane Position. This means a system engineer who wants to trace the failure logic associated with their system function can't just follow the functional decomposition.

The situation is further complicated when we have a system function (such as a fuel shut off valve) whose functional deviance can lead to several different hazards. In the case of our rotate variable stator vane function above, the fault logic for VSV: High Speed Stop (HSS) is also associated with the same functional block as VSV: LSS, even though their respective hazards are concerned with different phases of flight. In practice, as a user descends the functional decomposition and attempts to associate fault tree fragments with functional blocks, you quickly build a list of associated fault tree fragments that belong to either VSV:HSS or VSV:LSS, and in some cases where base events are shared, to both.

Unfortunately in PTC Windchill Modeller the context of a fault tree fragment cannot be shown in a menu item and it is difficult to know which fault logic diagram feeds into which from the list of associated diagrams. If we accept the utility of linking fault logic to functional models as a worthwhile objective, we need a way to correlate branches of the fault logic model to their corresponding functional blocks, regardless of how the decomposition is organized or whether the fault logic ignores functional boundaries.

The original idea behind MBSE was for all modeling and analysis to be done using a single data repository, thus avoiding engineers using different versions of specifications and divergent designs. However, achieving alignment with the system model requires not just a restructuring of the fault logic to match the functional decomposition, we also need a way to organize and group the fault logic models, as the parts that correspond to the functional specification are only fragments of a much larger, richer failure model. Without some guiding principle of organization, the semantically shallow associated diagram hyperlink defeats some of the gains made by bringing the system and safety models into closer alignment, as it could result in confusion about which parts of the failure logic are associated with which function. In order to gain the benefits of a single model data source, we needed a better approach to tie together fault logic and functional decompositions.

5. How to bind fault logic to design specification

We have already covered the removal of failure logic for functional redundancy so that we could focus the failure model on functional failures or deviations. But as explained above, having two separate model views (both held in SysML) was confusing for users to navigate and associate. It made future scripting for model verification more complex and didn't solve the issues of having to maintain two models in parallel. The linkage between system specification (activity diagrams in this case) and fault tree fragments was no more than a hyperlink association.

The SysML safety profile on the ECOSIStem small engine project took a different approach. It started by providing support for the FHA, both in terms of capture and export of the analysis into tables. The decision was made to embed the FHA within the system design and stakeholder requirements documents at an early stage and unlike the SysML fault tree profile for UltraFan, there was no requirement to support existing tool chains or being limited to activity diagrams. This gave the safety team freedom to create a simple profile based on integrating safety analysis features into functional specification documents on internal block definition diagrams (ibds). The profile defines relationships between hazards, failure modes, functional failures / deviations and some high level failure logic. At the FHA level some fault logic is modeled (mostly to give context to the hazard), and fault propagation is represented by attaching functional failures to the ports on function block properties. For reasons of space, we have decided only to include a representative

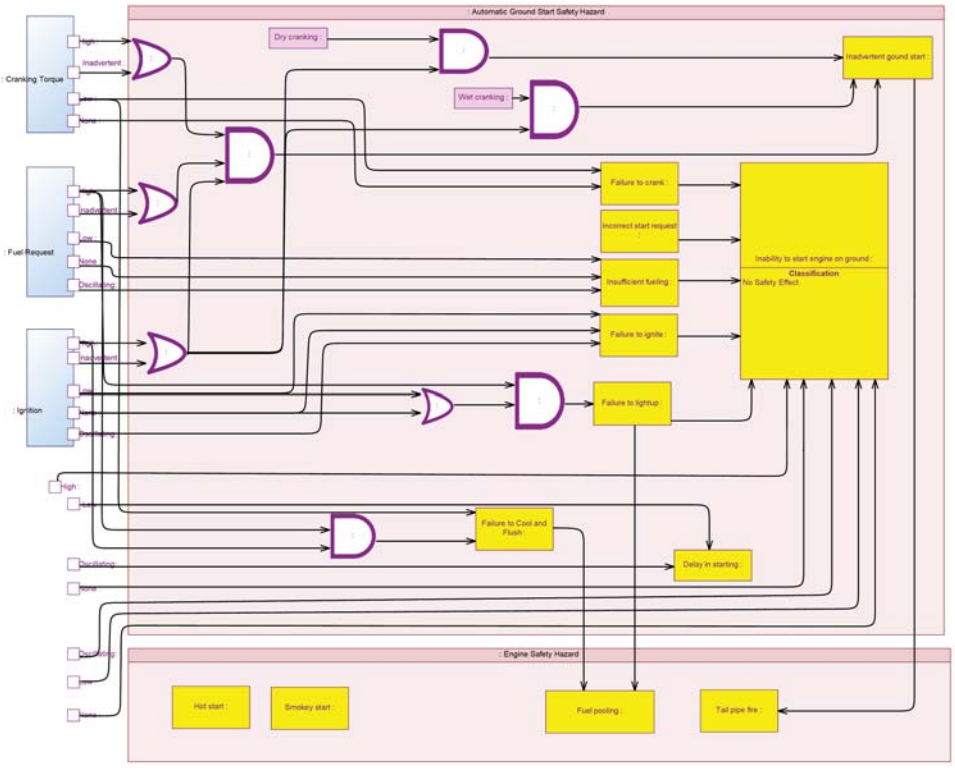

Fig. 4. An early snapshot of the FHA for Automatic Ground Start

example of part of the FHA (see Fig 4) covering automatic ground start. There is a level above this that links with the use case stakeholder requirements and hazard log. However, our focus here is on the lower level functional specification documents, and how we integrate the fault logic with these, eventually tracing up to the FHA.

Most system designs start with a description of the system architecture that is relatively abstract and which becomes more detailed towards the physical implementation. In ECOSIStem, this is done using logical and physical architectural layers. The FHA starts at the high level logical views, but continues the fault propagation through the logical decompositions into the physical architecture and functional grouping of specific hardware components. Although the original FHA safety profile wasn't intended to model fault propagation down to the level of hardware units, and it contains none of the specialist support for certain fault tree analyses such as engine dispatch, at this stage during its development we are more concerned with the primary requirement that the fault logic can be tightly bound to the design documents. This will restrict what can be modeled, and we discuss some of the shortfalls of the approach in Section 6.

A typical fault logic model for the Servovalve component specified the VSV's physical architecture is shown in Fig 5. We have omitted the property tags that define unique codes for each failure mode from the Failure Mode Effects and Analysis (FMEA) database (this is maintained externally to the SysML model). What is important to note here is that this ibd for the Servovalve block property is not one created by the safety team. This is part of the physical component architecture defined by system design. The safety engineer takes an instance of this block and enhances it with fault logic information. The ports are also pre-defined by system design, and the functional failures that propagate between blocks are added in a similar fashion. This process ensures that the fault logic is created within a function block provided by system design using the correct port definitions. If changes are made by the system engineers to the port or block specification, those changes will show up on the fault logic block instance as well.

Scripts can be used to verify that all failure modes input to a gate or port, and that all functional failures trace to a top level hazard. This mechanism for storing the fault logic associated

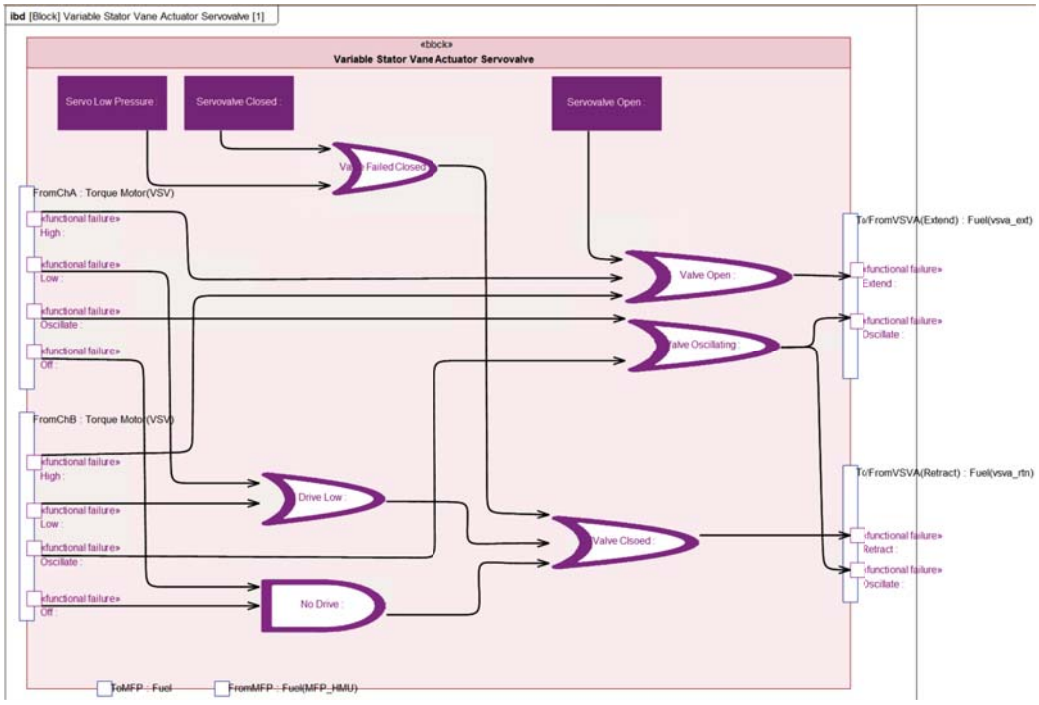

Fig. 5. Internal fault logic of the Variable Vane Servovalve ibd, showing both control channels inputs, OR / AND gates and failure modes (purple blocks providing inputs to gates). The input and output ports map functional failures between component blocks.

with functional deviations flowing down from the FHA means that the system engineers have access to the fault logic and failure modes of any function they wish to inspect. They can visually trace where the functional failures come from and propagate to (although we hope to extend this functionality with scripts) and there is none of the difficulty associated with trying to navigate two models that decompose in different ways with respect to functional boundaries. Here, the function block is the 'wrapper' for its associated fault logic.

6. Quality assessment

While this gives an overview of the modeling side, clearly the fault logic still has to be exported for analysis by a tool like RWB FaultTree+. The need for export made us consider the possibility that parts of the fault logic model should either sit outside the system's functional decomposition (such as library components for common hardware) or outside the SysML model altogether (which is the case for some third party component suppliers who provide the fault logic and failure modes for their products). Given that there will always be parts of the failure model that would not be held in SysML alongside the functional decomposition, the export of SysML fault logic model will be part of process of 'stitching together' fault logic from different sources before the full model is imported into the analytical tool. For an engine FADEC like that at Rolls-Royce, the vast majority of fault logic is concerned with capturing and tracing the functional deviations highlighted in the FHA. What remains (Common Cause Analysis, zonal analysis, non-functional hazards such fuel leaks or lightning strike) can be incorporated at a later stage in SysML if there is an advantage in doing so. However, as we have demonstrated, trying to incorporate or associate two differently layered models alongside one another can result in confusion for users, an increase in cost and no discernible improvement in quality.

Although we cannot provide hard evidence, we are convinced that embedding the majority of the FADEC fault logic inside the system's functional specification will bring significant developmental improvements. The single biggest cost factor for safety analysis is having to rework fault logic models and analysis due to changes in system specification. If MBSE can provide a tightly integrated environment for MBSA, then much of the duplicated effort or rework can be avoided. There are also 'soft' benefits associated with greater vis-

ibility for system engineers to check failure modes and fault logic associated with their functional specification. For example, changes to data acquisition and validation can have immediate impact on the failure modes of components that use them and this is much easier to see when the functional deviations are part of the system design blocks.

7. Conclusions and future work

Our ongoing work to model fault logic within the SysML system specification is far from complete. We started out demonstrating that existing fault tree models could be captured in SysML, but failure models have a much wider scope than functional decompositions and we failed to take into account the practical considerations of model alignment. By starting from the system design blocks and limiting the fault logic to the propagation of functional failures, we were able to achieve much tighter integration with system design and start to realize some of the benefits of MBSE for safety analysis. There remains part of the failure model that can't be incorporated into the functional decomposition in a useful way. This is not to say it can't be modeled in SysML, but consideration needs to be given to the value and cost of moving it into that format, given the likelihood that the full failure model will be composed from different sources anyway before analysis. We hope to explore modeling non-functional hazards such as fire risk or leaks in fuel-hydraulic systems, but these won't be as tightly bound to the system specification and the cost savings for re-working the safety analysis are less obvious. Overall, our aim is to find an 'optimal' integration of safety processes into SysML, so that quality is improved and errors between systems and safety engineers are reduced, but which avoids adding process complexity of little value.

References

Biggs, G., T. Juknevicius, A. Armonas, and K. Post (2018, 07). Integrating Safety and Reliability Analysis into MBSE: overview of the new proposed OMG standard. *INCOSE International Symposium 28*, 1322–1336.

Boiteau, M., Y. Dutuit, A. Rauzy, and J.-P. Signoret (2006). The AltaRica data-flow language in use: modeling of production availability of a multi-state system. *Reliability Engineering and System Safety 91*(7), 747–755.

Clegg, K., M. Li, D. Stamp, A. Grigg, and J. McDermid (2019a). A SysML Profile for Fault Trees - Linking Safety Models to System Design. In Romanovsky (Ed.), *SAFECOMP 2019*, Volume 11698 of *Lecture Notes in Computer Science*, pp. 85–93. Springer.

Clegg, K., M. Li, D. Stamp, A. Grigg, and J. McDermid (2019b). Integrating Existing Safety Analyses into SysML. In Papadopoulos (Ed.), *Model-Based Safety and Assessment IMBSA 2019*, Volume 11842 of *Lecture Notes in Computer Science*, pp. 63–77. Springer.

David, P., V. Idasiak, and F. Kratz (2009, 09). Automating the synthesis of AltaRica Data-Flow models from SysML. *ESREL 2009 Vol 1*.

IEC 61025:2006 (2006, August). IEC 61025: Fault tree analysis (FTA). Standard, International Electrotechnical Commission, Geneva.

Joshi, A. and M. P. E. Heimdahl (2005). Model-based Safety Analysis of Simulink Models Using SCADE Design Verifier. In *SAFECOMP'05*, pp. 122–135. Springer-Verlag.

Lisagor, O., T. Kelly, and R. Niu (2011). Model-based safety assessment: Review of the discipline and its challenges. In *The Proceedings of 2011 9th International Conference on Reliability, Maintainability and Safety*, pp. 625–632.

Micouin, P. (2014). *Safety Engineering*, Chapter 10, pp. 183–212. John Wiley & Sons, Ltd.

Rauzy, A. and C. Blériot-Fabre (2014, Nov). Model-Based Safety Assessment: Rational and trends. In *MECATRONICS 2014- Tokyo*, pp. 1–10.

Schallert, C. (2017). Automated safety analysis by minimal path set detection for multi-domain object-oriented models. *Mathematical and Computer Modelling of Dynamical Systems 23*(3), 341–360.

Shao, N., Zhang, S., and Liang, H. (2017). Model-based safety analysis of a control system using Simulink and Simscape extended models. *MATEC Web Conf. 139*, 00219.

Sorokos, I., Y. Papadopoulos, L. Azevedo, D. Parker, and M. Walker (2015). Automating Allocation of Development Assurance Levels: an extension to HiP-HOP. *IFAC-PapersOnLine 48*(7), 9 – 14. 5th IFAC International Workshop on Dependable Control of Discrete Systems.

Tanaka, N., H. Yomiya, and K. Ogawa (2020). A method to support the accountability of safety cases by integrating safety analysis and model-based design. In Casimiro (Ed.), *SAFECOMP 2020 Workshops*, Volume 12235 of *Lecture Notes in Computer Science*, pp. 23–35. Springer.

Acknowledgement

This work is funded as part of Innovate UK's ENCASE (Enabling Novel Controls and Advanced Sensors for Engines) project.