
Proceedings Paper:
Clegg, Kester Dean orcid.org/0000-0002-4484-3291, McDermid, John Alexander orcid.org/0000-0003-4745-4272, Grigg, Alan et al. (1 more author) (2019) Integrating Existing Safety Analyses into SysML. In: Papadopoulos, Y, Aslansefat, K and Katsaros, P, (eds.) Model-Based Safety and Assessment (IMBSA) 2019:Lecture Notes in Computer Science. Springer Nature , pp. 63-77.

https://doi.org/10.1007/978-3-030-32872-6_5

Integrating Existing Safety Analyses into SysML

Kester Clegg[1], Mole Li[2], David Stamp[2], Alan Grigg[2], and John McDermid[1]

[1] University of York, United Kingdom, YO10 5DD
{kester.clegg,john.mcdermid}@york.ac.uk
[2] Rolls-Royce (Controls) PLC, Derby, United Kingdom
{mole.li,alan.grigg,david.stamp}@rolls-royce.com

Abstract. Migrating systems and safety engineering (often with legacy processes and certified tools) towards a model based systems engineering (MBSE) environment is a socio-technical problem. Establishing a common conceptual framework requires agreement on modelling artefacts and the integration of existing tool chains to minimise disruption. We discuss our experience integrating a SysML Safety Profile to model fault trees but which has the prerequisite requirement to continue the analysis of those models by existing tools. We demonstrate a lightweight profile that minimally captures the fault logic for a Rolls-Royce gas turbine engine controller and provides specific in-house extensions for both fault tree and engine dispatch analysis by exporting model entities and relationships from the SysML fault trees. During integration we realised a more fundamental need to reconcile the systems engineers functional view with the safety engineers focus on failure modes and fault logic in order to maximimse the longer term benefits of MBSE development.

Keywords: SysML · Fault Tree Analysis · Failure Modes

1 Introduction

Systems engineers have traditionally used separate models of the system functions from those used for safety analysis. Part of this stems from the need to consider the system from a functional perspective on one hand and on the other hand how it will fail. As failures frequently cut across functional boundaries and model very different things, system and safety models can be difficult to reconcile and verify for consistency. While it can be argued that maintaining two models from a single set of system specifications can act as an independent check that the system will behave as expected under failure, the differences between the system and safety models is often a source of inefficiency and misinterpretation.

In this paper we document our experience that trying to reconcile system and safety perspectives is not simply a question of sharing a single data repository captured in a modelling language such as SysML (Systems Modelling Language). Support for different perspectives requires alignment not only of artefacts, but of how the system should be modelled to gain a common understanding across

the company's engineers. The context of this work is as part of Rolls-Royce's UltraFan engine demonstrator program,[3] which has elected to trial SysML during system development.

1.1 Paper structure

§1.2 gives some background to safety critical and systems modelling using SysML perspective and previous work. §1.3 describes the specific requirements for the ENCASE (Enabling Novel Controls and Advanced Sensors for Engines) project. §2 looks at our implementation of the fault tree SysML profile. §3 details our bespoke SysML profile that provides support for modelling fault logic both within SysML and through the use of export scripts to existing fault tree analysis tools. §4 gives a summary of engine dispatch analysis.[4] The issue of gradually introducing MBSE through integration with the existing analytical toolchain is covered in §5. In §6 we discuss some of the problems and solutions to reconcile different modelling viewpoints with respect to the functional specification and derived safety requirements. Finally §7 concludes our experience and outlines the work going forward.

1.2 Background and previous work

Model Based Systems Engineering (MBSE) 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. Various flavours of MBSE have been proposed over the last two decades [12] that targeted the needs of systems development. The references listed here are mostly pertinent to safety critical civil aerospace development as to cover all topic domains within MBSE would require a more extensive review. However, even within the more restricted remit of safety critical aerospace systems and safety modelling there is a wide variety of approaches, with many based around particular languages (AltaRica[3],[5],[13], SCADE/Lustre[10]), or around a modelling environment such as Matlab Simulink [15] in combination with other tools, such as HiP-Hops [16] or physical simulation environments such as Modelica or Simscape ([14],[15]. The decision to adopt SysML as the modelling language for UltraFan was taken prior to our work starting on ENCASE.

SysML is an extension of the Unified Modelling Language (UML) that focuses on systems modelling. SysML supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems.[5] However, 'support' in this sense is intended to mean a well-defined specification

[3] Part of Innovate UK's ENCASE (Enabling Novel Controls and Advanced Sensors for Engines) project.

[4] Dispatch refers to the engine's ability to carry a fault for given time before maintenance action is taken.

[5] This paper refers to the current Object Modelling Group (OMG) SysML v1.5, not the upcoming 2.0 standard. See http://www.omgsysml.org/

to describe the system, so that development and analysis can be performed using tools that take their data from a single model repository. This allows existing (perhaps certified) tools to be used provided that a means to export the data from the repository into a format the tool can use is made available. In order to do that, an input method must be provided that allows the critical information and knowledge capture of both system and safety concerns. Unfortunately while a graphical interface for system modelling is widely supported by tool vendors for SysML, a similar environment for safety analysts to model fault logic is rarely provided. Fault logic is typically modelled using a graphical representation of logic gates that traces the fault from base event to effect and which can contain additional information, such as failure rate, dispatch information and descriptive failure modes. A typical example is shown in Fig. 1 and the technique is defined in standards like IEC 61025 [9].

In 2017 the OMG issued a Request for Proposals 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, such as hazards and the harms they may cause, model reliability analyses, including Fault Tree Analysis (FTA) and Failure Mode and Effects Analysis (FMEA), and use structured argument notation to organise the model and specify assurance cases"[2]. As part of this, an early profile for Fault Tree Analysis (FTA) and Failure Mode and Effects Analysis (FMEA) has been developed and published [2] and is likely to form part of SysML 2.0. However, while the new profile is moving in the right direction, it isn't sufficiently defined to be adopted for use on the development of UltraFan within Rolls-Royce and neither is it likely to support the specific requirements for Rolls-Royce to model engine dispatch availability. Our work attempts to bridge this current gap in SysML capability by providing a bespoke SysML profile to support Rolls-Royce's Fault Tree and Time Limited Dispatch (TLD) analyses.

1.3 ENCASE project

ENCASE's initial starting point to model fault trees in SysML was an early paper from the National Aeronautics and Space Administration (NASA)'s Jet Propulsion Laboratory on fault protection modelling, which captured fault logic using UML (Unified modelling language) activity diagrams [6]. We investigated the potential of this approach but found issues with it. Firstly, while it is possible to model OR logic gates on activity diagrams using nodes, there is no provision for AND gate representation. The nature of the system redundancy provided by a dual channel FADEC (Full Authority Digital Engine Control)[11] means that modelling fault logic requires the use of AND gates (due to the possibility of the same function on both channels failing). Secondly activity diagrams were never intended to model fault trees, and trying to use them for that purpose inevitably brings compromises. At Rolls-Royce Controls, system engineers are already using activity diagrams to model system functions and incorporating safety model artefacts like fault logic gates using activity diagram notation would cause confusion and the potential for misunderstood syntax / semantics.

Furthermore, at least as implemented in PTC's Integrity Manager, activities on activity diagrams become Call Behaviour Actions, which semantically seems an over specification for fault logic that is minimally expressed as set of fault propagation paths containing logic gates. Although there are other potential diagram types within SysML, none offer specific support for fault tree analysis and we decided we could best meet our needs by creating a bespoke diagram type.

There is also recent work investigating the formal translation of activity diagrams in UML / SysML to fault trees [7]. While this is a rigorous method, that entails a one to one correspondence between the two models, at this stage in the ENCASE project a more pragmatic approach is required due to the variety of ways engineers model activities. For example there are parts of activity diagrams, such as Join Nodes, that are semantically ambiguous and can be used / interpreted differently by users which would make automated translation difficult. However there is a more fundamental problem with attempting a direct translation, in that traditional fault logic models often contain quite abstract failure modes that will not have a corresponding entity in a functional model. For example system engineers may model functional behaviour that mitigates against a known hazard, but they are unlikely to model the *loss* of that function and its effect on the system. Therefore the fault tree may contain fault logic that cannot be linked to or directly translated from entities within activity diagrams. It may be possible to do a partially automated translation if both models were carefully constructed to reflect the same functional hierarchy and channel implementation. We discuss this possibility in more detail in §6.

The primary practical concern for the safety team was that the SysML fault tree models should be capable of modelling the system fault logic as it had been done historically and exporting it in a format where it could be analysed by their existing tools such as FaultTree+ (part of Isograph's Reliability Workbench suite). Their requirements were that the graphical user interface should be as close as possible to FaultTree+ and that the information kept in the SysML model should be the minimum required to export the fault logic for analysis. This made adapting some existing approaches, such as Component Based Fault Trees [1] unsuitable as they were felt to be too complex for what was needed, despite the requirement for modular fault trees. Being able to compose sub fault trees that can be joined to existing branches of fault logic for specific forms of analysis, such as time limited dispatch analysis, is supported via transfer gates and in this respect mirrors the functionality offered by FaultTree+.

2 Implementation

The current modelling environment for UltraFan is provided by PTC's Integrity Modeler (formerly Artisan) using SysML extended to aid efficient modelling of gas turbine controllers. A typical screenshot is shown in Fig. 2 and to date the software is mostly used to capture system specification through activity diagrams. In the left hand panel, below the activity diagrams in the package hierarchy can be seen the fault tree structures. Our initial trials showed that

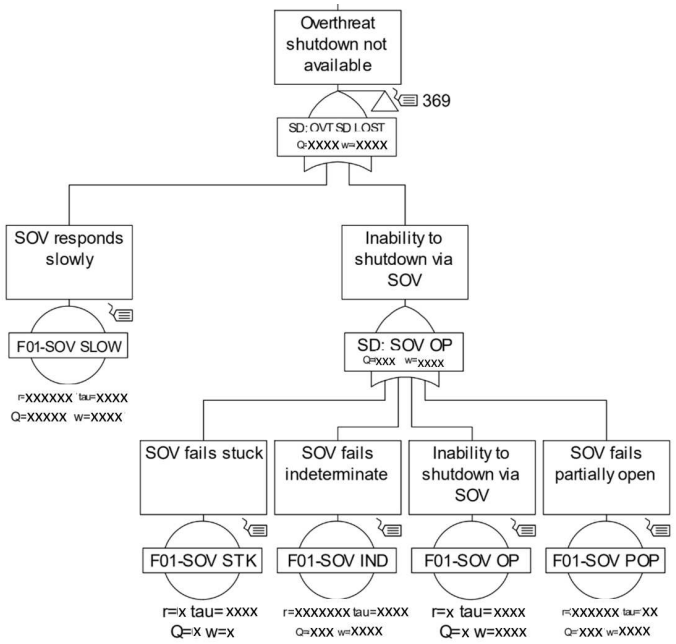

Fig. 1. Lower level of a fault tree showing base events with FMES (Failure Mode and Effects Summary) identifiers (unique names) as rendered by RWB's FaultTree+.

there are some user interface issues with very large fault trees being represented in a 'file browser' type format, as the user can quickly get lost scrolling through hundreds of gates. However, there are tools within PTC IM that allow a quick search between entities on the fault tree diagrams and their location with the package browser.

In order to bridge the gap between traditional safety engineering that uses separate models from the system engineers' models, and in a similar spirit to the OMG RFP mentioned earlier, we have drafted the first stage of our Model Based Safety Assurance (MBSA) profile that will in time allow the full integration of safety analysis models with existing system models. Our profile remains a work in progress and this part is sufficient to start to migrate the existing fault tree models into the SysML repository. Similar to SysML extensions in UML, the proposed Fault Tree Profile reuses a subset of UML 2.5 and provides a bespoke diagram type (an extension of structured diagram) and additional gate definitions to aid specific types of Fault Tree and dispatch analysis for Rolls-Royce. The initial version of our profile is detailed in [4] and the profile's entities and linkages are described in detail there. However, we have since released a new version with substantive changes, in particular the removal of 'failure modes' as a first class entity, due to issues with the user interface and ease of export to analytical tools (see §3).

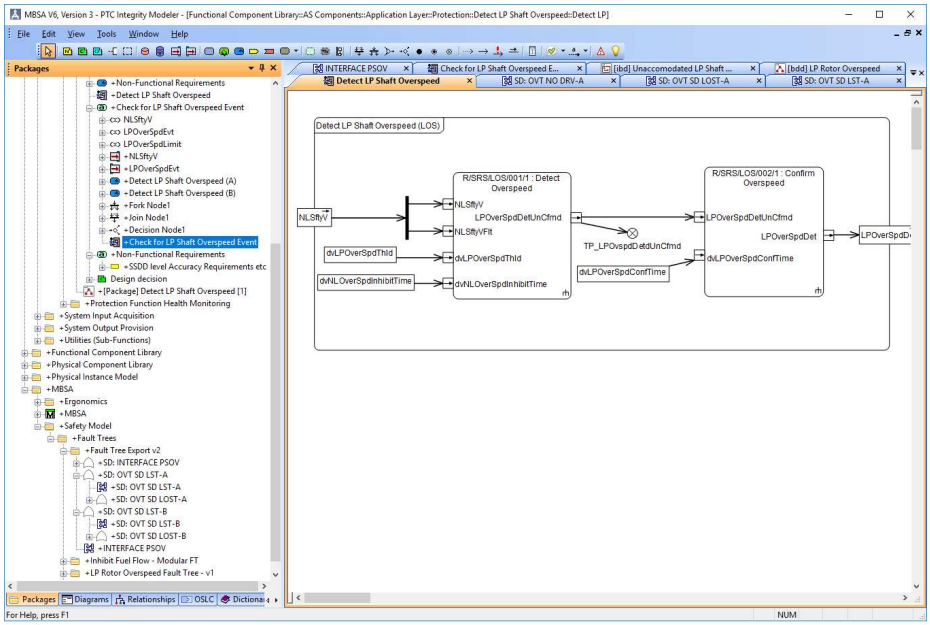

Fig. 2. Example of activity diagram modelled in PTC Integrity Modeler (formerly Artisan).

The main aim of the profile was to capture the minimum information needed to accurately export the fault logic to FaultTree+ and to ensure that a single specification was used to drive both safety and systems modelling. Using as lightweight a profile as possible means much of the FMES (Failure Modes and Effects Summary) base event and dispatch information does not need to be kept in the SysML model.[6] Instead, the events and gates have unique identifiers that is sufficient for the information associated with them to be extracted from the FMES database. The reason for this is that the FMES is quite large (>3K rows with many columns) and there has to be an explicit case made for bringing that information into the SysML model where it is less easy to keep it maintained and checked.

Therefore it is easiest when a new analysis is to be run to extract the summary failure rate data directly from the databases, while keeping the fault propogation logic, base and dispatch events within the SysML model. This is in keeping with our belief that the SysML model represents a knowledge repository, whereas the FMECA and FMES databases are designed to handle, import and export

[6] The FMES is a derived summary of the Failure mode, effects and criticality analysis (FMECA) database (>25K rows) which is maintained with the latest failure rates.

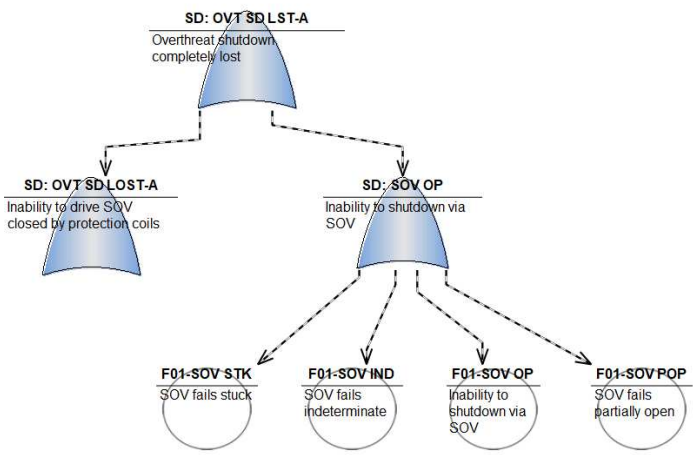

Fig. 3. Lower level of H01 (this is a top level hazard for turbine overspeed) fault tree showing OR gates and base events with FMES identifiers, as rendered by PTC Integrity Modeler using our Fault Tree SysML profile.

large amounts of data efficiently and are able to interface with a wide range of analytical tools. Fig. 3 shows an example fault tree modelled using our profile in PTC Integrity Modeler. Removing the FMES data (which is not used by the safety analysts when modelling the fault logic — it is added by FaultTree+ by combining the failure rates of base events) gives a much cleaner interface. The gate descriptions or 'failure modes' are tags on both events and gates.

3 Changes to the Previous Fault Tree Profile for SysML

Version 1 of our profile is shown in [4] which describes in detail the profile's entities. However, due to user experience studies, we have had to make some fundamental changes to the profile and have further extended it with transfer gates, null gates and dispatch events (see Fig. 4) in order to accommodate the types of analysis for engine dispatchability that are specific to Rolls-Royce (civil aerospace). As engine dispatch analysis is a complex topic, we give a short summary in §4.

Our changes to the profile centre around the removal of 'failure modes' as a first class entity that could be linked to other parts of the SysML specification (see Fig. 3 that shows using gate descriptions as 'failure modes'). The motivation for having them as first class entities in the profile was to enable a more flexible traceability to derived safety requirements and to enable verification checks so that each failure mode was associated with a function and every function was associated with at least one failure mode. Unfortunately, user tests revealed that users would often 'copy and paste' failure mode instances when modelling dual channel functions (instead of creating unique failure modes for each channel).

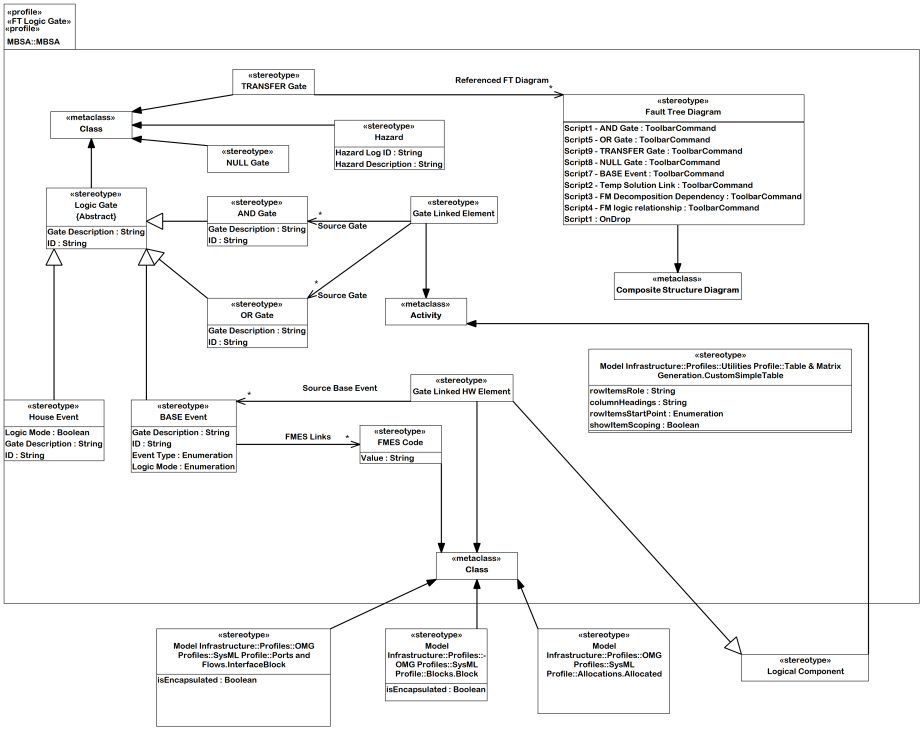

Fig. 4. Meta model of the proposed Fault Tree Profile, which will form part of a larger MBSA profile. The Fault Tree Diagram scripts are not part of the profile but serve to recreate a familiar user interface for safety analysts in PTC Integrity Modeller (PTC IM). The export script is not shown.

The effect of this was that the model would link that failure mode instance to both logic gates, so that it would end up with two inputs (one from each channel).[7] While the fault tree diagrams looked fine to the user, on exporting the fault tree logic to Reliability Workbench, it was realised that these failure modes had the wrong number of inputs to the next gate. Although this issue could perhaps have been addressed by suitable user training, it was felt that this was not particularly user friendly due to the linked inputs being effectively 'hidden' from the user (i.e. the additional links were not visible on the fault tree diagram).

The solution was to remove failure modes and instead consider them as 'human readable' descriptions of the logic gates in the fault tree. This simplified the model parsing for export and removed some of the 'clutter' of the fault tree dia-

[7] As explained later, a single gate with two inputs from either channel is possible where both channels access the same hardware component and therefore share the same fault logic. However, that is a specific case and is definitely not correct in the case of a duplicate control function running on each channel.

grams. As most gates have a unique identification with respect to their channel, this reduced the possibility of the user creating 'hidden' links in the model by using an existing gate defined for another channel. There are exceptions to this, as there are hardware components that both channels use (such as the fuel shut-off valve) and for which there is a single set of associated base events and fault logic. In this case, the user must take care to define the gates and events that represent the shared hardware above the split in the fault tree branch that models the implementation of a specific channel's fault logic, so that both channels can have access to an instance of the gate or event on their respective branches of the fault tree. This ability to model the shared hardware for either channel or repeated instances of hardware is particularly important for common cause analysis.

3.1 Additional extensions to the profile

The rationale for creating a bespoke fault tree SysML profile is so that in-house modelling techniques and practices can be maintained with as little disruption or additional training as possible as the transition is made to MBSE. In the case of Rolls-Royce, a specific gate called a *TRANSFER gate* is used for linking sub trees (often stored in separate files) to branches of an existing fault tree. This means that sub fault trees that model shared system resources (such as hardware or network buses) can be built up into libraries and added to models as required. This has the advantage that if change needs to be made to a sub tree, it can be made once and the change will be reflected wherever that sub tree is used.

The second type is a variant of a base event termed a *House event* and this is used to model the presence of dispatch faults in certain configurations needed for dispatch analysis (see next section). House events as implemented in FaultTree+ are base events except that their logic mode is restricted to either true or false. Selecting them to TRUE (logic mode) incorporates the event into the analysis. Selecting the house event to FALSE removes it from the analysis. House events can be modelled under an OR gate or an AND gate dependent upon the system effect being modelled.

At Rolls-Royce Controls *NULL gates* are sometimes used above a house event as a type of neutral interface. NULL gates do nothing except pass the input onward, however they are more flexible than a direct input from a base event if changes are needed, as NULL gates can take an input another gate or subtree, whereas a base or house event cannot. House events are primarily of interest for engine dispatch analysis in order to satisfy the requirements of CS-E 1030 and the process is briefly described in the following section.[8]

4 Modelling Time Limited Dispatch

A FADEC system is designed to be fault tolerant so that many single faults lead to loss of redundancy rather than functionality. This enables airlines to operate

[8] See [8] for a detailed discussion on the Time Limited Dispatch requirements for more-electric gas turbine engines with respect to CS-E 1030.

engines with faults in the control system until a convenient place and time of repair is reached. At the end of each flight the on-condition maintenance ensures that the system provides a record of known faults (if any) and determines whether the faults within the system are sufficient to prohibit dispatch. If departure is allowed with known faults then in many cases a time limit is set for the repair to be carried out.

With respect to base event models and time limited dispatch, there are two types of maintenance policy:

- On-condition maintenance requires that a fault be repaired within a fixed period of time after a fault is detected. This is modelled using the time at risk model with all faults conservatively assumed repaired at the end of the allowable period.
- Fixed interval maintenance only repairs faults at one of a number of scheduled maintenance slots. When a fault is detected it is repaired at the next slot. This is modelled with the 'dormant' model with repair rate set to zero and the inspection interval set to the period between maintenance slots. Note, zero repair time is used since the safety models only consider flight time and the repairs effectively take no flight time (no repairs in flight!) regardless of the actual repair time on-ground.

Certain events do not have an associated control systems dispatch period and instead have an immediate effect. These are modelled as Do Not Dispatch (DND) faults and may be designated as initiating events. A number of event groups have been defined and these include an event group for each of the main exposure periods (i.e. DND, Short Time Dispatch (STD), Long Time Dispatch (LTD), Unlimited Dispatch (ULD), and Dormant) along with additional groups for any exposure periods that may arise that do not fall within the main categories. In general the dispatch period used for a base event should be that set by the dispatch status generated by that fault when it occurs while the system is in a 'full-up configuration'. This strategy gives the correct results for one or two fault cut sets. Issues may occur with three fault cut sets. The dispatch information is not kept in the SysML model, in keeping with our principle that the profile should be as lightweight as possible and that information is easier to maintain and manage via the FMES and FMECA databases.

There are three main aspects to the Fault Tree Analysis for Time limited Dispatch (TLD):

1. Fleet average rate calculation.
2. Specific rates for individual dispatchable configurations.
3. Cut set analysis to demonstrate that no hazardous event can be caused by a single control system fault in any dispatchable configuration.

The first is covered by setting exposure periods for base events. The second is covered by modelling dispatchable configurations using House Events. The house events are added for each Dispatchable Fault (DF) identified in the dispatch summary. These are added both to the individual main and sub-models, and

their logic mode set FALSE. Each house event in turn is selected to TRUE and the model run, giving results for each dispatchable configuration. The third also uses the dispatchable fault house events. It involves setting their logic mode to basic to ensure they appear in cut sets and then examining the cut sets for all Hazardous events to ensure that there are no cut sets where both a dispatchable fault and a single control system fault occur. If such a cut set existed it would indicate that there is a dispatchable configuration where a single control system fault results in a hazardous event.

In FaultTree+ a base event can take three values for its logical mode — basic, true or false. A house event can only take either true or false, and therefore it is either part of tree as a dispatch fault that is 'on' or excluded as an input. This is a tool specific extension we include in profile so that export to FaultTree+ will support existing methods of analysis. If a different tool was being used for analysis, it would be possible to create a profile that extended the base event class to create a dispatchable fault that contained a simple boolean tag to indicate if it should be part of dispatchable configuration for analysis. The ability to extend profiles in this way to match the export needs of specific analytical tools is one of the great advantages of SysML.

In order to demonstrate compliance to the requirements of CS-E 1030 it is required to calculate the top event rates of the various hazards in each of the declared dispatchable configurations. To facilitate this analysis requires the addition of a number of dispatchable events to the fault trees in order to model degraded configurations. Previously this analysis was performed manually through the use of NULL gates that 'switched' house events to TRUE. However, as we discuss in the following section, thanks to the extensive automation interface provided with PTC IM, it will in future be possible to largely automate this configuration of this task using export scripts.

5 Using scripts to integrate analytical tools

Bringing together different engineering perspectives under the SysML umbrella is complicated by the established traditions and processes for those disciplines. Engineers get used to particular tool idiosyncrasies and work arounds, but more fundamentally they establish a level of trust through methods of working with the tools. In order that the migration process towards MBSE retains these trusted methods, accomodating existing analytical tools is essential. Fortunately, many tools allow import and export of data into spreadsheets or CSV (comma separated values) files. The current environment for modelling SysML at Rolls-Royce Controls is PTC's Integrity Manager.

PTC's IM comes with an extensive automation interface that can use Visual Basic (VB) scripts to provide customisations to the user interface, to edit and change models and to export data via formats such Microsoft's Excel database. This facility has been of great benefit when creating the user interface for our bespoke fault tree diagram. For example, it was possible to replicate to a large extent the look and feel of FaultTree+, so that users could use familiar icons

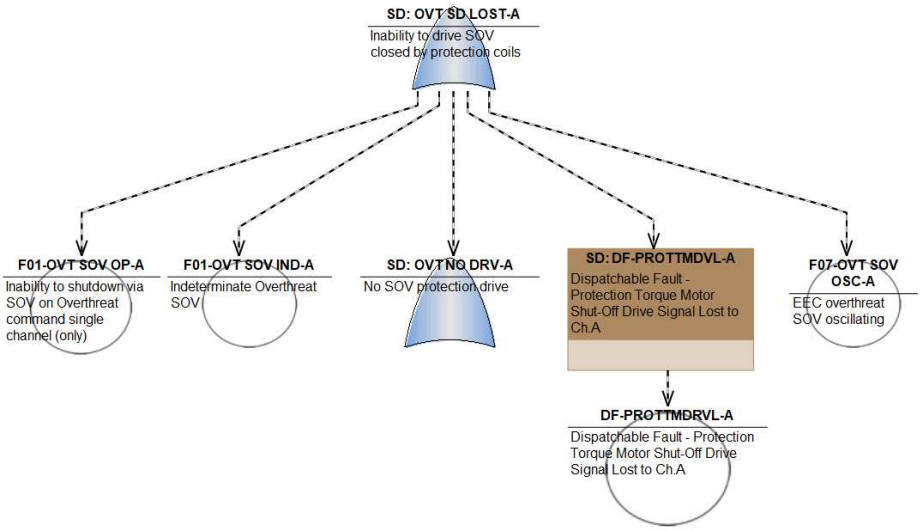

Fig. 5. Dispatch event shown with NULL gate. Export scripts can identify these events through Rolls-Royce's naming convention and enable them by setting their logical mode to TRUE and then exporting the fault logic for that dispatch configuration.

and graphic symbols in their diagrams. An example of additional functionality is to enable them to create new branches of the fault tree by double clicking on a gate with no inputs. This opens a new fault tree diagram if one does not already exist and the user can use the same gate instance on the new diagram to help readability. Furthermore the use of scripts can limit the types of action allowed on fault tree diagrams by prohibiting the wrong 'links' between entities or warn the user if the cardinality between entities is exceeded, it is even possible to perform look ups to match gate names against the FMES database. But the real value of scripts in the SysML model is allow exports to analytical tools.

5.1 Exporting fault tree logic

By choosing a minimal capture of fault logic for the fault tree profile, the information required to extract for import into FaultTree+ is relatively easy to obtain. FaultTree+ requires fault logic imports to summarise two worksheets, one for base events and one for the gates and their inputs. Due to the automation interface, the data repository can easily parse all classes belonging to a package. In our case, the gates and events are extensions of classes and so these can be filtered from the data dictionary. The complexity comes from maintaining and identifying the dependents and dependees for each gate. The dependent relationship is the output of that gate into another gate. The dependee relationship is the inputs to that gate from other gates or events. A typical output is shown in Fig. 6. Once the fault logic has been imported, the analysis can be

	A	B	C	D	E	
1	Name	Type	Description	Input 0 Type and Name	Input 1 Type and Name	Input 2 T
2	SD: OVT SD LOST-A	OR	Inability to drive SOV closed by protection coils	E:F01-OVT SOV OP-A	E:F01-OVT SOV IND-A	G:SD: OVT
3	INT: SOVHSS OP-A	TRANSFER	SOV drive High Side Switch open due to EEC fault affecting multiple signals			
4	SD: OVT SOV OP-A	OR	Inability to energise SOV protection drive	E:F07-OVT SOV DRV OFF-A	G:INT: SOVHSS OP-A	G:INT: SO
5	SD: OVT NO DRV-A	OR	No SOV protection drive	G:CON: PSOV INV-A	G:SD: OVT SOV OP-A	
6	CON: PSOV INV-A	TRANSFER	protection SOV drive signal lost due to harness / connector faults			
7	SD: OVT SD LOST-B	OR	Inability to drive SOV closed by protection coils	E:F01-OVT SOV OP-B	E:F01-OVT SOV IND-B	G:SD: OVT
8	INT: SOVHSS OP-B	TRANSFER	SOV drive High Side Switch open due to EEC fault affecting multiple signals			
9	SD: OVT SOV OP-B	OR	Inability to energise SOV protection drive	E:F07-OVT SOV DRV OFF-B	G:INT: SOVHSS OP-B	G:INT: SO
10	SD: OVT NO DRV-B	OR	No SOV protection drive	G:CON: PSOV INV-B	G:SD: OVT SOV OP-B	
11	CON: PSOV INV-B	TRANSFER	protection SOV drive signal lost due to harness / connector faults			
12	SD: SOV OP	OR	Inability to shutdown via SOV	E:F01-SOV STK	E:F01-SOV IND	E:F01-SO
13	SD: DF-PROTTMDVL-A	NULL	Dispatchable Fault - Protection Torque Motor Shut-Off Drive Signal Lost to Ch.A	E:DF-PROTTMDRVL-A		
14	SD: DF-PROTTMDRVL-B	NULL	Dispatchable Fault - Protection Torque Motor Shut-Off Drive Signal Lost to Ch.B	E:DF-PROTTMDRVL-B		
15	INT: SOVLSS OP-A	TRANSFER	SOV drive Low Side Switch open due to EEC fault affecting multiple signals			
16	INT: SOVLSS OP-B	TRANSFER	SOV drive Low Side Switch open due to EEC fault affecting multiple signals			
17	SD: INTERFACE PSOV	OR	Protection SOV	G:SD: OVT SD LST-A	G:SD: OVT SD LST-B	
18	SD: OVT SD LST-A	OR	Overthreat shutdown completely lost	G:SD: SOV OP	G:SD: OVT SD LOST-A	
19	SD: OVT SD LST-B	OR	Overthreat shutdown completely lost	G:SD: SOV OP	G:SD: OVT SD LOST-B	
20						

Fig. 6. Export of gate logic to Excel worksheet. The fault logic is represented by the inputs to each gate (up to 17, including whether it came from an event or another gate) and the dependent gate (the gate that receives the output). The failure modes in the previous profile were replaced as descriptions of the gate. Although 25 columns in the worksheet for the gates are specified (and a similar number for events), the gate and base event unique IDs are sufficient for extracting additional information from the FMES / FMECA as needed to analyse dispatch configurations.

run as usual. The probability and exposure data behind the fault logic remains in the FMES and FMECA databases and can be extracted as needed into the SysML model or FaultTree+.

5.2 Automating the dispatch analysis

As explained in §4, dispatch analysis is carried out by selecting house events and setting them to TRUE in the fault tree and running the analysis. To date this has been a manual task, and quite a substantial one given the combination of dispatch configurations and events. However, now that information is in the SysML model, it can be parsed by scripts that can generate a set of dispatch configurations for export into FaultTree+. The dispatch status of each event is maintained in the FMES (Failure Mode and Effects Summary) database and can be extracted to create a list of dispatch configurations. The script loops through each configuration, and selectively generates an export containing each enabled dispatch event integrated into the fault logic as needed. These are then passed on to FaultTree+ and the analysis run as usual. Being able to automate the generation of fault logic for the different dispatch configurations represents a considerable saving of man hours.

6 Alignment of safety and system models

Advocates of MBSE are quick to point out the improved fidelity and efficiency of maintaining a single development model. However, as safety engineers have traditionally modelled their understanding of the system's fault logic with respect to a hazard independently of other system models, some abstract failure conditions may have little obvious connection to system functions. In such cases,

a realignment and reassessment of failure modes may be necessary. For example safety engineers often model a system with respect to its redundancy and mitigation against a hazard, thus an analysis for a dual channel control system might query why the mitigation provided by the redundant channel has failed in addition to the channel in control. Contrast this with the system engineer's perspective, which is to consider an engine protection feature in its abstract specification first, then its implementation and finally how it is implemented on a respective channel. In the move towards using a single SysML model for all system development and analysis, little benefit is going to be gained unless concept and viewpoints on the system share a common understanding and reference points. For example, rather than the top level fault logic models starting by querying channel redundancy, they could follow where possible the functional hierarchy provided by the system engineers and instead consider redundancy at the level of channel implementation. Fault trees are often "richer" than system models in that they may have to include physical or external factors that lie outside the system's functional specification but are required to understand how that function could fail. In such cases it can seem there is little correspondence between the system and safety models, but such differences can be overcome by ensuring a flexible profile that allows links to hardware and activity models alike from the fault logic. Visibility of the associated fault logic for functions can then be provided to the system's engineers without the unnecessary addition of unrelated events that are present in the full fault tree.

7 Conclusions

In this paper we have sought to identify some of the benefits and problems when migrating system and safety modelling under the MBSE SysML umbrella. Through the use of lightweight bespoke profiles and user interface scripts, analysts gain familiar means to input their models into the SysML repository. The short term benefits are that analysts are able to continue with tried and trusted analytical methods by exporting data to existing tools, with the additional benefit of potentially time saving auto-generation of certain analyses such as dispatch configurations. However, longer term benefit requires a more significant shift towards a common understanding of how the system should be specified and analysed, so that system and safety engineers can cross-reference each others models and ensure better traceability from derived safety requirements. Looking longer term still, we can expect to see the OMG's SysML 2.0 safety profile solidify to give stricter semantics within meta-models, leading to the possibility that large parts of the fault logic could be auto-generated from system functions and hardware models.

References

1. Adler, R., Domis, D., Höfig, K., Kemmann, S., Kuhn, T., Schwinn, J.P., Trapp, M.: Integration of component fault trees into the uml. In: Dingel, J., Solberg, A.

(eds.) Models in Software Engineering. pp. 312–327. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
2. Biggs, G., Juknevicius, T., Armonas, A., Post, K.: Integrating Safety and Reliability Analysis into MBSE: overview of the new proposed OMG standard. INCOSE International Symposium **28**, 1322–1336 (07 2018)
3. Boiteau, M., Dutuit, Y., Rauzy, A., Signoret, J.P.: 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 (2006), https://EconPapers.repec.org/RePEc:eee:reensy:v:91:y:2006:i:7:p:747-755
4. Clegg, K., Li, M., Grigg, A., Stamp, D., McDermid, J.: A SysML Profile for Fault Trees — linking safety models to system design. In: SAFECOMP Proceedings 2019. Springer (*yet to be published*)
5. David, P., Idasiak, V., Kratz, F.: Automating the synthesis of AltaRica Data-Flow models from SysML. Proceedings of ESREL 2009 **1** (09 2009)
6. Day, J., Murray, A., Meakin, P.: Toward a model-based approach to flight system fault protection. In: Aerospace Conference, 2012 IEEE. pp. 1–17. IEEE (2012)
7. Dickerson, C.E., Roslan, R., Ji, S.: A Formal Transformation Method for Automated Fault Tree Generation From a UML Activity Model. IEEE Transactions on Reliability **67**(3), 1219–1236 (Sep 2018)
8. Fletcher, S., Norman, P., Galloway, S., Burt, G.: Impact of engine certification standards on the design requirements of More-Electric Engine electrical system architectures. SAE International Journal of Aerospace **7**(1), 24–34 (September 2014)
9. IEC 61025: Fault tree analysis (FTA). Standard, International Electrotechnical Commission, Geneva, CH (August 2006)
10. Joshi, A., Heimdahl, M.P.E.: Model-based Safety Analysis of Simulink Models Using SCADE Design Verifier. In: Proceedings of the 24th International Conference on Computer Safety, Reliability, and Security. pp. 122–135. SAFECOMP'05, Springer-Verlag, Berlin, Heidelberg (2005)
11. Li, M., Batmaz, F., Guan, L., Grigg, A., Ingham, M., Bull, P.: Model-based systems engineering with requirements variability for embedded real-time systems. In: 2015 IEEE International Model-Driven Requirements Engineering Workshop (MoDRE). pp. 1–10 (Aug 2015). https://doi.org/10.1109/MoDRE.2015.7343874
12. Lisagor, O., Kelly, T., Niu, R.: 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 (June 2011). https://doi.org/10.1109/ICRMS.2011.5979344
13. Rauzy, A., Blriot-Fabre, C.: Model-Based Safety Assessment: Rational and trends. In: 2014 10th France-Japan/ 8th Europe-Asia Congress on Mecatronics (MECATRONICS2014- Tokyo). pp. 1–10 (Nov 2014). https://doi.org/10.1109/MECATRONICS.2014.7018626
14. Schallert, C.: 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 (2017). https://doi.org/10.1080/13873954.2017.1298624
15. Shao, N., Zhang, S., Liang, H.: Model-based safety analysis of a control system using Simulink and Simscape extended models. MATEC Web Conf. **139**, 00219 (2017). https://doi.org/10.1051/matecconf/201713900219
16. Sorokos, I., Papadopoulos, Y., Azevedo, L., Parker, D., Walker, M.: Automating Allocation of Development Assurance Levels: an extension to HiP-HOP. IFAC-PapersOnLine **48**(7), 9 – 14 (2015), 5th IFAC International Workshop on Dependable Control of Discrete Systems