Abstract—Systems of systems (SoSs) are vulnerable to faults, for example arising as a result of the distribution and independence of their constituent systems. Our previous work has presented an initial framework for reasoning about faults and fault-tolerant design within an SoS at the architectural level, using a simple example of a single failure. In this paper we present a motivating example of an SoS within which failures of constituent systems (CSs) may lead to a degraded or partial service for the SoS. We discuss a possible extension to our earlier framework to allow an architectural engineer to reason about the fault tolerance of an SoS where CSs contribute negatively or positively towards a goal or an SoS-level failure, and where multiple SoS faults may interact to affect SoS service quality.

I. INTRODUCTION

In a system of systems (SoS), the components of the overall system are themselves independent systems. An SoS is typically large, complex, distributed, and operating in a domain with high requirements for safety, reliability and accuracy. As such, there are particular demands on the SoS engineer to consider fault tolerance carefully. For example, connectivity problems are a risk and the constituent systems (CSs) may withdraw from the SoS arbitrarily. Independently-owned CSs may evolve without taking into account the needs of the SoS, or providing advance warning. CSs may also balk at the need to implement possibly costly recovery processes for faults introduced by third parties. For these reasons, we believe that there is a need for quality tools and methods to support reasoning about the SoS at the architectural level. The COMPASS project has developed a Fault Modelling Architectural Framework (FMAF) that provides a systematic approach to capturing fault tolerance aspects of SoSs, described in previous publications [1], [2], [3].

In this paper we present a new case study that presents a different approach to failure, in that a ‘failure’ is interpreted as an increasingly degraded level of service to which faults contribute. Using this as a motivating example, the contribution of this paper is to demonstrate how to apply the COMPASS FMAF to support reasoning about degraded level of SoS service and the representation of failures as events or as states. We suggest some extensions to the FMAF to accomplish this. Following our earlier examples, we use SysML [4] to present our models.

The rest of this paper is laid out as follows: Section II summarises some previous work on architectural approaches to fault modelling and Section III briefly introduces the Fault Modelling Architectural Framework we use. Section IV introduces our case study. We describe here how we incorporate notions of degraded service into the FMAF using some extensions to the FMAF. Finally Section V presents our discussion and conclusions.

II. RELATED WORK

Fault-tolerant architectures have been widely studied in the literature (e.g., see [5]). Architectural approaches are particularly useful for modelling fault tolerance within an SoS at an early stage – for example, for identifying CSs that may be responsible for introducing, detecting and recovering from various faults (the independence of the CSs can be a problem for ensuring a fault-tolerant service is delivered, when functionality is composed from interactions between CSs). Architectural approaches have been demonstrated previously in related fields, such as modelling of erroneous behaviour in embedded systems [6], [7], analysis of dependable complex physical systems [8] and verification of safety requirements in safety-critical embedded control systems [9]. Other work explores architectural approaches for modeling non-functional requirements (e.g., an extension to SysML to allow goals to be refined into sub-goals [10]).

We rely on an existing dependability taxonomy [11], but we make some slight adaptations to suit an SoS context. We define a failure as a deviation of the service provided by an SoS from the expected (correct) behaviour. An error is defined as the SoS state that can lead to its subsequent service failure. The adjudged or hypothesized cause of an error is called a fault. Point of view is important for SoS modelling: a failure from the point of view of a single CS becomes a fault from the point of view of the SoS.

III. FAULT MODELLING ARCHITECTURAL FRAMEWORK

The Fault Modelling Architectural Framework (FMAF) has been developed by the COMPASS project to provide a systematic approach to capturing fault tolerance and dependability aspects of SoSs. Initial work on the FMAF has been described previously [1], [2]. The FMAF particularly provides support for: definition of faults, errors and failures; the identification of the causal chains of dependability threats, of the CSs and their interfaces, and of erroneous behaviour/recovery processes; behaviour description of processes in the presence of faults and recovery process. The FMAF incorporates a total of eight
different viewpoints, used to explore and present faults and fault-tolerant concepts within an SoS. We present only a short selection of viewpoints here; a complete list can be found in [1], [2], [3].

IV. CASE STUDY

Our case study is supplied by West Consulting, and is extracted from a Traffic Management System (TMS) that controls the inter-urban road network in the Netherlands. The TMS collects information about current conditions and traffic flow and takes actions to achieve some predefined global goals regarding traffic behaviour. The goals of a TMS are complex and include: improving efficiency of the road network; ensuring road safety; reducing the negative impact of problems such as accidents and blockages; and reducing environmental impact (e.g., controlling pollution in certain areas). This involves the collaboration of a wide variety of CSs, as well as cooperation with external entities such as TMSs operated by third parties in adjacent regions. The CSs within the SoS include:

- Systems of traffic monitors – varied devices for monitoring current traffic
- Systems of actuators – devices that can influence traffic via signs and signals
- A set of regional Traffic Control Centres (TCCs) that analyse region-wide data from monitors, select traffic management strategies and issue appropriate instructions to actuators
- The Traffic Information Centre (TIC), that collects nationwide traffic data, and makes it available to nationwide service providers. The TIC occasionally implements a national traffic management strategy

We present here a model that considers a subset of TMS functionality, focusing especially on ramp metering functionality that is delivered by the TMS.

A ramp meter system (RMS) is situated on the access ramp used to access inter-urban highways. The RMS employs two-phase (red and green) traffic lights to control the rate at which vehicles join the highway. Figure 1 presents an informal representation of an RMS. The RMS prevents bottlenecks from being formed when many vehicles join a major road, improves vehicle distribution by breaking up ‘platoons’, and can reduce accidents caused by high speed merges. An RMS typically has access to data about traffic in its own immediate vicinity (this differs to the regional control centre (TCC) which has access to region-wide traffic data). The RMS operates in one of several modes:

- Fixed-time mode, with fixed-length red/green phrases
- Adaptive mode, which responds to current traffic conditions, and can be further divided into:
  - Responsive mode, where the RMS makes ramp-metering decisions using local data only.
  - Collaborative mode, where the TCC makes ramp-metering decisions using regional data, overriding the local RMS decisions.

The collaborative mode is selected by the regional TCC if it detects that a regionally-implemented RMS strategy is needed. Collaborative mode is selected occasionally and is not the default behaviour for most RMSs.

A. Nominal behaviour

In this section, we present a SysML model of the TMS nominal (non-faulty) behaviour. We concentrate on RMS functionality specifically, omitting models of other TMS functions (of which there are many). We have therefore simplified the SoS to include:

- a regional TCC
- a distributed collection of data monitors employed by the TCC. We abstract away the implementation details of traffic monitors; this function may be implemented by a variety of different devices, and by more than one CS
- three CSs that influence traffic behaviour:
  - a distributed system of RMSs
  - a separate distributed system of configurable speed limit panels
  - a distributed system of dynamic route information panels

We have modelled the TCC as cycling alternately through two phases: a phase of data gathering and analysis, followed by a phase of implementing strategies based on the analysis. Figure 2 presents a SysML activity diagram depicting the initial data gathering phase. In this phase, the regional TCC requests data from the traffic monitoring system(s). The TCC has access to (in this case) three actuator CSs that can help to influence traffic behaviour: a system of RMSs; a system of configurable speed limits; and a system of dynamic information panels. We assume that each of these actuator systems maintains information about its current state (e.g., current availability, existence of faults etc.) We represent this information in Figure 2 using one data store for each CS to hold the configuration data. After receiving traffic reports from the traffic monitoring CSs, the TCC collects details about the current configuration of each of the actuator CSs from the data stores. It then makes decisions about appropriate traffic management strategies, and writes the details of the selected strategies back into the relevant data stores.

Next the TCC moves to the subsequent phase of operation, which is to implement the selected strategies to regulate traffic behaviour (shown in Figure 3), by delivering instructions to each of the relevant CSs on the desired behaviour, before returning to the data gathering phase. Figure 3 depicts the TCC forwarding instructions to the dynamic route information system and the configurable speed limit system; however the
RMS has a more complex process, where it may be gathering local data and making decisions in isolation, or receiving instructions from the TCC instead. The RMS regularly collects and analyses data about the local area and, when necessary, selects a responsive mode of operation, in which it responds to local conditions instead of adopting fixed durations for the red/green phases of the traffic lights. This allows finer control over the rate at which vehicles join the TMS. Data gathering and analysis continues throughout this process and the RMS will revert to the simpler fixed time mode if appropriate.

Figures 4 and 5 depict the RMS: Figure 4 shows a high level perspective of the RMS, which is capable of operating in an isolated mode or a collaborative mode; and details of the RMS’s isolated mode (which may be either fixed-time, or responsive) are shown in Figure 5.

At any stage an instruction from the TCC may be received, requesting that the RMS move into collaborative mode (shown on Figure 4 as a signal received by the RMS, and the interrupted RMS processes contained in a SysML interruptible region). In this case, the RMS has a simpler process, as it should now admit vehicles on a schedule decided by the TCC and does not need to study local conditions or determine the appropriate mode. On receiving a signal to end collaborative mode, the RMS resumes the analysis of local conditions.

B. FMAF views of the TMS

In this section we present viewpoints defined in the FMAF to describe the RMS functions in terms of fault tolerance. We implement our FMAF views in SysML, making use of a stereotyping profile created as part of the COMPASS project, to define FMAF entities and their properties.

1) Redundancy in the TMS: Figure 6 depicts a Fault Tolerant Structure View (FTSV) of the TMS. In this view we show the possible redundancy that is available within the TMS, by identifying which CSs may be able to provide similar services. For example, the Variable speed limit and Dynamic route info systems provide some redundancy for the RMS. We identify this with the providesRedundancyFor relationship. However, we provide some qualification for this relationship; the three systems all influence traffic behaviour to some degree, but they do not have identical effects. If the RMS is unavailable, then the TCC may be able to reduce the resulting negative impact on traffic flow to a limited extent by altering the speed limit to compensate, but the overall net effect on traffic flow, and the potential side-effects of each system, are not exactly the same. We represent this by adding an extra property (a tag in SysML) to the providesRedundancyFor dependency relationship to specify whether this is a total or partial redundancy. In this case, we suggest that the Variable speed limit and Dynamic route info systems can only provide a partial redundancy for the RMS.

2) Failures of the TMS: In Figure 7 we present an initial Fault/Error/Failure View of the case study, which allows us to
state the failures of CSs (these are faults at the SoS level), the SoS error states, and the SoS-level failures in which we are interested. The TMS has a number of high-level requirements and goals, but the one towards which the RMS contributes is a requirement for the TMS to deliver ‘optimal’ traffic flow for the given conditions\(^2\). Whilst this is not necessarily a safety-critical goal for the TMS, it does have substantial impact on, for example, regional or national economic performance (by reducing the time and costs of transporting goods and people), as well as reducing pollution levels and fuel burned.

Different faults within the RMS may result in either of these SoS-level failures arising. The two failures are closely related and affect each other; for example, preventing vehicles from accessing the TMS is one (suboptimal) strategy that could help to ensure that traffic flow within the SoS is smooth and ‘optimal’ for vehicles already in the system. On the other hand, if traffic flow within the SoS is not optimal and traffic jams arise, then the ramps will become blocked with vehicles waiting to enter. Because these failures are not independent, there’s a possibility that recovery strategies adopted by the TMS to recover from one failure may inadvertently create conditions to lead to the other failure arising. For this reason, we assign each failure a priority, to allow an engineer to consider a variety of recovery processes and strategies, and, where multiple strategies are available, select the one with the ‘optimal’ effect. We implement this by adding an extra property (SysML ‘tag’) to the ‘Failure’ entity to capture this.

3) Faults of the TMS: Alongside the two SoS-level failures, Figure 7 presents the erroneous SoS states that may lead to the failures, and the faults\(^3\) that could cause those errors. The SoS-level faults that could arise within the RMS include:

1) Lights stuck on green, or no lights at all
2) RMS fails to adopt collaborative mode when instructed
3) Lights stuck on red
4) RMS fails to exit collaborative mode when instructed
5) The RMS calculates an incorrect rate for vehicles to be admitted
6) Access ramp blocked
7) RMS is uncontactable (e.g., due to a power failure or persistent communications problem)

These two faults result in similar SoS erroneous states as Faults 1-5 caused by the RMS: an access ramp that is blocked (e.g., by a broken-down vehicle or some debris) will have a similar result to Fault 3 (lights stuck on red); whilst an uncontactable RMS will not be entering (Fault 2) or exiting (Fault 4) collaborative mode when instructed. Although Faults 6 and 7 are not RMS failures in themselves, we do want to make sure that the SoS can recover from environmental problems and so we may want to model the detection of these

---

\(^2\)We do not attempt to define ‘optimal’ traffic flow here (a major piece of work in its own right), but assume that the TMS has a notion of what is optimal at any one time, taking into account some predefined key priorities and goals.

\(^3\)As mentioned earlier, faults at the SoS level are considered to be failures of the constituent systems
problems and their recovery processes. There are other SoS-level faults introduced by other CSs which may affect the TMS’s delivery of RMS functionality (e.g., a problem with the TCC, causing it to wrongly select/deselect collaborative mode), but due to space limitations we concentrate here on faults introduced by the RMS CS only.

4) Example Fault-Error-Failure chain in the TMS: Taking two example faults of the TMS, as identified in Figure 7, an RMS stuck on red (Fault 3) and access ramp blocked (Fault 6), these lead to different errors of the TMS. Depending on driver behaviour, vehicles may be admitted at a suboptimal rate (Error 1), or no vehicles are admitted (Error 2). These errors may lead to two TMS failures; suboptimal traffic flow in the TMS (Failure 1) and vehicles are unable to enter the TMS (Failure 2) respectively. We can consider the priorities assigned to the two failure events when selecting an appropriate recovery strategy. For example, if a ramp has been blocked or stuck on red there are likely to be vehicles waiting to join the TMS, and potentially traffic will be backed up in the surrounding area, as vehicles are unable to exit the surrounding local roads onto the major highway via the ramp. Once the blockage is cleared, there is a choice of recovery strategies:

- Adopt the usual optimal rate for admitting vehicles. This prioritises preserving the flow of traffic on the major highway.
- Admit vehicles more quickly than usual, to clear the backlog of vehicles waiting on the ramp and alleviate congestion in the surrounding area. This prioritises clearing the ramp, but vehicles on the highway will have to reduce speed to accommodate the new influx of vehicles, and blockages on the major highway may arise.

Assigning priorities to failure events can support decision-making when selecting the best recovery strategy. Given a full description of the fault/error/failure propagation, in the next section, we consider reasoning about Fault 2 in more detail using the FMAF.

C. Analysing Fault 2

In this section, we provide a worked example of the use of the FMAF to reason about one of the faults identified in Figure 7. We use Fault 2 (RMS fails to adopt collaborative mode when instructed) as an example. If the RMS fails to switch into collaborative mode when required, and no recovery action is taken, then the RMS will most likely be admitting a suboptimal rate of vehicles to the TMS, because the rate of vehicles admitted during the collaborative phase is determined at a regional level using traffic data unavailable to the local RMS. This will eventually lead to the failure event ‘SoS fails to deliver optimal traffic flow for current conditions’. There are two main causes this fault:

1) There may be a problem with the communications.
2) The RMS fails to respond correctly because of an internal fault.

In this paper, we concentrate on the second of these for our fault modelling; representing a failure of the RMS.

4Our models here do not include a notion of a ‘communications’ CS that carries signals between CSs. However, if this is a potential source of faults we advocate representing it as a CS in SoS models so that the impact of its failure can be analysed (for an example of this strategy see [1]).

detection of the SoS error state will be the same, regardless of the CS responsible for activating the fault, but the recovery processes will differ so we need to be able to determine which CS activated the fault. Modelling each cause of the fault allows detection and separate recovery processes to be designed.

Figure 8 presents a Threats Chain View for Fault 2, which illustrates how the original fault affects the SoS, beginning as a CS-level failure (a fault within the SoS), progressing to an SoS error state and finally the failure event. In this case we show that the fault arises within the RMS and propagates to become visible to an observer at the boundary of the system as suboptimal traffic flow. The CSs involved with activating the fault and/or detecting it are identified at each stage (the Threats Chain View differs for Fault 2 if the fault is activated by poor communications).

Fig. 8. Threats Chain View (Fault 2)

We next produce a Fault Activation View (FAV) for Fault 2, shown in Figure 9. This model shows the SoS-level process for initiating collaborative mode. The instruction from the TCC to initiate collaborative mode is received by the RMS, but the process is interrupted by the activation of Fault 2 before the RMS can enact the switch into the new mode. As a result of the fault activation, the RMS continues to operate in isolated mode. This represents an erroneous behaviour, and left unchecked can lead to the failure event of suboptimal flow. We view the error (of continuing in isolated mode) as a continuous state, rather than a single event. Figure 9 represents this as two parallel processes following fault activation: the RMS enters a continuous isolated process, and in parallel the failure event may arise.

Figure 9 also signals the start of the appropriate recovery process (shown in detail in Figure 10), initiated by the TCC. The recovery for this fault is two-pronged. The responsibility for detection lies with the TCC, which should be able to gather traffic data sufficient to estimate the rates at which the individual ramps are admitting vehicles to the system, and compare this to the expected rate. The TCC takes two actions: a report of the problem is logged, to prompt maintenance teams to inspect the RMS equipment for faults. To cope with the short-term situation, the TCC updates its RMS data store with the information that one of the RMSs is admitting vehicles at an incorrect rate, and then returns to its cycle of analysing data and making decisions. The data analysis and traffic management strategies made by the TCC now take into the account the malfunctioning RMS.
This recovery process represents extra small tasks undertaken by the TCC during its data-gathering and analysis phase, which was represented in Figure 2; we can modify Figure 2 to include this level of detail if necessary.

V. CONCLUSION

In this paper, we use the FMAF to model fault tolerance in an industry case study – the TMS. As most TMSs feature constituents that have been in use for decades alongside very recent additions, they may exhibit different features, error detection and failure recovery strategies. The views provided by the FMAF allow TMS engineers to plan and reason about recovery strategies that are necessary in order to, for example, incorporate a brand new CS into the system, by identifying where responsibility lies for detection. The FMAF provides a structured process for capturing and analysing where responsibilities lie within the SoS, as well as a system for capturing rationale behind process design in a readable and structured format. This paper presents the first use of the FMAF for modelling degraded states in SoSs. Future work will consider a larger industrial-scale example with multiple interacting degraded states to evaluate the FMAF viewpoints and their ability to model such complexities.

In using the FMAF, we motivate several extensions to the FMAF, particularly: SoS failures modelled with a sliding scale of degraded service; failures assigned a priority and a severity to aid recovery planning; provision of partial redundancy for a CS; and modelling of erroneous behaviour as a continuous state.

There are several areas for future and ongoing work in this area of SoS fault tolerance, including: modelling remaining faults in the TMS case study, incorporating the newly identified extensions into the FMAF framework definition, and the development of tools and techniques to support linking between architectural models and fault analysis tools (such as HiP-HOPS\(^5\)) and formal verification techniques.

ACKNOWLEDGMENTS

The work presented here is supported by the EU Framework 7 Integrated Project ‘Comprehensive Modelling for Advanced Systems of Systems’ (COMPASS, Grant Agreement 287829): http://www.compass-research.eu. The authors are grateful to staff at West Consulting for their help and input.

REFERENCES


\(^5\)http://hip-hops.eu