Formal analysis of safety and security requirements of critical systems supported by an extended STPA methodology.

Giles Howard*, Michael Butler†, John Colley§, Vladimiro Sassone‡
Department of Electronics and Computer Science,
University of Southampton,
United Kingdom
giles.howard@soton.ac.uk, †mjb@ecs.soton.ac.uk, §j.l.colley@ecs.soton.ac.uk, ‡vsassone@soton.ac.uk

***Abstract*—Cyber-physical systems represent an engineering challenge due to their safety and security concerns, particularly those systems involved in critical infrastructure which require some of the highest standards of safety, availability, integrity and security. The complexity of these systems makes the identification and analysis of safety and security requirements challenging. In this paper, we present a methodology for identifying and formally analysing safety and security requirements, based on the STPA methodology and combined with modelling, traceability and formal verification through use of the Event-B formal method. Our STPA approach is then leveraged to generate 'critical requirements' to mitigate against undesirable system states, which are then subsequently translated into constraints on an Event-B representation of the system. The Rodin toolset then allows us to demonstrate that these critical requirements fully mitigate against the undesirable system states and therefore provide automated verification of the critical requirements.**

***Index Terms*—System analysis and design, systems modeling, cyber-physical systems, formal verification**

1. Introduction

Cyber-physical systems - those systems with a real-world physical component and "whose operations are monitored, coordinated, controlled and integrated by a computing and communication core" [1] - are increasingly finding a place in infrastructure across the world, as well as in domains such as vehicle design and healthcare. While these cyber-physical systems present opportunities to monitor and manipulate the physical world through computation [2], there exists within this growing field a distinct need to ensure safety and security of such systems which are a synergy of software and hardware wherein any failure modes within the system may have profound effects on anything from the success of a continued medical treatment to the survival of a plane full of passengers. It is therefore clear that this new category of computing systems require an extensive regime of design and formal assurance support to ensure that these opportunities are fully embraced while still maintaining the safety and security of this class of systems.

This represents a challenge - particularly on the side of the security of these systems - where historically cyber-security researchers are said to have not considered how attacks affect the estimation and control algorithms and ultimately, how attacks affect the physical world [3]. It is therefore clear that there is a requirement for a robust methodology to ensure that the inseparable need for both safety and security within cyber-physical systems is addressed.

One possibility for addressing the need to ensure both security and safety of a system is through the use of formal method techniques. Cai et al. concisely describe the potential benefits of formal methods as follows:

> Formal methods is an effective way to improve the quality of large-scale software and reduce the cost of software development... formal methods also enable accountable validation and verification of the resulting system, which reduces or even eliminates possible defects in the software and thus better maintains safety of the system [4].

This view is reinforced in the recently-published guidelines for systems security engineering [5], which states that "Security, like safety and other system quality properties, is an emergent property of a system." Since this emergent behaviour "cannot be predicted through analysis at any level simpler than the system as a whole" [6], the sheer complexity of this analysis means that traditional testing methods alone are insufficient for full and proper verification - formal methods are needed. In this paper, we propose a methodology combining a technique for analysing systems for hazardous states and producing system level constraints, which we will then translate into a formal method model.

2. State of the art

2.1. Historical accident models

Traditional models for understanding the origin of accidents within systems originate with Heinrich and his Domino Model [7]. This focus on direct causation (as represented by the linear nature of the metaphorical dominoes falling one by one) simplified the notion of an accident to

a simple series of failures wherein the prevention of one failure could prevent the accident in its entirety.

Heinrich's model inspired many refinements and evolutions of his original contribution - Bird and Loftus [8], as well as Adams [9] both presented modernisations of the Domino Model in order to maintain its relevance during the latter half of the 20th century through attempting to rebalance the contribution of organisational and regulatory failings to the linear chain. It also encouraged the development of the 'Swiss Cheese' model presented by Reason [10] which took an alternative approach of viewing an accident as a result of the chance alignment of multiple failures in a "perfect storm" but still viewed the occurrence of an accident as a linear process once more - this time, as slices of swiss cheese.

Criticisms have been levelled at these accident models for simplifying the nature of accidents too much and failing to take into account the sheer number of contributing factors in most accidents [7], [11].

2.2. STPA and derivative methodologies

2.2.1. STAMP and STPA. Systems-Theoretic Process Analysis and its associated accident model - Systems-Theoretic Accident Model and Processes - depart significantly from the historical model of Heinrich and his contemporaries. In this accident model, *accidents* are not the result of a linear chain of events but instead are viewed as resulting from *inadequate enforcement* of constraints on system behaviour [12]. Safety is therefore an emergent property which is a result of adequate control of the system through sufficiently complete constraints on system behaviour. Failure to adequately constrain the system can result in the system entering an unsafe state and thus a certainty of causing an accident when worst-case environmental conditions occur.

The STAMP model also views that systems can be adequately explained through interrelated components that are kept in equilibrium through feedback loops and control actions. Controllers are viewed as having a 'process model' which represents their understanding of the current state of the process they are controlling and they update this with information coming from a 'feedback' or sensor channel (which may correlate to multiple actual sensors in the system). They then 'act' on the process in response through some form of actuator. Any given system therefore consists of a multitude of these controllers - acting on each other and their individual processes to carry out the functions of the system. Safety therefore is ensuring that these individual controllers maintain their process within a set of constraints, as well as ensuring that these constraints cover all potential states of the underlying process. There is an additional requirement that the process model for each controller is also detailed enough and that the sensor channels can read - to a sufficient granularity - the current process state and relay this information.

A "safe" system contains some number of controllers which all have a thorough understanding of the underlying process that they are controlling, and which can also react to maintain the state of their processes within some set of constraints. Safety is therefore a result of emergent behaviour and accidents often are a result of more than a single action but multiple failings across a series of control loops at a variety of levels - this is the fundamental difference between a systems-theory-based approach as compared to the existing linear/casual chain-of-events models embodied by the Domino Model and its derivatives [11], [13].

2.2.2. Methodologies derived from the STAMP/STPA approach. One of the first substantial pieces of work to build on STPA was the STPA-Sec methodology proposed by Young [14]. This extension of the existing STPA methodology focuses tightly on the notion of "mission assurance" and the security elements of a system under analysis. Through manipulating the terminology used in the analysis, it is possible to utilise STPA-based analysis to derive security vulnerabilities which may lead to loss states. The result of the analysis is a set of constraints which can then be used to iterate the system concept and design towards a more secure design.

Further methodologies - such as STPA-Priv - generalise the STPA approach once again, but this time towards privacy-based concerns. This is primarily realised through modification of the terminology once more to allow the privacy issues to be highlighted through the analysis - the notion of a 'loss' is instead substituted for 'adverse consequences' as this terminology is better understood in the field of privacy engineering [15].

STPA-SafeSec is an additional methodology developed to address perceived shortcomings in the STPA-Sec methodology, which was criticised for viewing security as an issue only in relation to safety. STPA-SafeSec seeks to improve the capabilities of STPA-Sec by aligning terminology and viewing security and safety as equal concerns when performing the analysis [16].

It is therefore clear that the STPA approach has demonstrated itself to be quite malleable in the hands of researchers and has permitted refinement and development in a variety of ways as demonstrated by the methodologies detailed in this section.

2.3. Formal methods and system design

There has been substantial use of formal methods within the context of system design within the literature. Formal methods are defined within the literature broadly as "mathematical techniques, often supported by tools, for developing software and hardware systems" [17]. Formal methods have been used in many contexts and have even been used by the likes of Amazon as part of their Web Services product to aid in its design and verification [18].

Of the many formal methods available in the literature, we focus on the Event-B formal method which takes a set-theoretic approach to modelling of systems [19] and has been utilised in modelling a number of case studies [20], [21].

Event-B has also been paired with the STPA methodology previously in a paper by Butler & Colley [22] wherein the artefacts generated from the STPA analysis were utilised in an Event-B model in order to leverage the automated proving and model checking capabilities available with this formal method and its associated tools. This analysis used the standard STPA technique and was therefore focused on identifying and demonstrating successful mitigation against unsafe system states within the resultant model.

3. Proposed methodology and process

We propose a methodology building on the STAMP/STPA model. The proposed methodology takes a lot of cues from the STPA-Sec and STPA-SafeSec methodologies, but aims to couple this with a more focused modelling approach.

The approach can be broadly summarised in a series of discrete steps, which will be expanded on at a later point in this paper. Before this is discussed, we devote a short amount of time to the specific nomenclature involved in the methodology, as well as the role of the Event-B formal method within the methodology, before describing the process of the methodology more thoroughly.

3.1. Nomenclature

Please see Table 1 for clarification on terms.

3.2. Role of the formal method

The usage of the Event-B formal method - and the associated Rodin toolset - aid the methodology by providing a more robust assurance that identified critical requirements actually mitigate against hazardous system behaviours.

This is due to the following:

1) The use of the Rodin tool for the Event-B formal method provides the ability to use interactive proving (and a variety of formal proof plugins) to demonstrate formally that the properties of the model hold or are unprovable in their current state. This can aid in improving the model (and thus the underlying system design) and also aid in improving the critical requirements themselves.
2) The use of model checking within Rodin to ensure there are no deadlock states which may be a risk when constraining system behaviour. Model checking can also provide counter-examples where there is a possible series of events which may undermine any given system constraints which can allow the analyst to consider development/refinements of each critical requirement to ensure that critical requirements cannot be undermined by a specific path of system behaviour.
3) Event-B supports the notion of refinement of a model as well as two types of decomposing models into sub-models which can then be composed to

TABLE 1. TERMINOLOGY USED WITHIN THE METHODOLOGY

Term	Meaning
Hazard	Hazards are unsafe or insecure system states which, when coupled with worst case environmental conditions, will result in a loss. The STPA family of methodologies makes the assumption that environmental conditions (being out of control of the system) will certainly align to represent the worst case and therefore focuses on preventing the system from entering into these hazardous system states. We utilise the term 'hazard' for both identified safety and security problem states in contrast with some existing work in the literature which uses 'vulnerability' for security states [14] - we use the same term in the interests of brevity.
Loss	A loss represents a failure of the system in fulfilling its declared purpose. All loses are deemed to be of equal importance and are derived from the system purpose. A loss may represent a security breach or an accident - the term "loss" is utilised due to its applicability in both security and safety domains.
System purpose	A sentence (or set of sentences) describing the overarching goals and responsibility of the system under analysis.
Critical Requirements	Constraints generated as a result of the modified STPA methodology proposed by this paper. These constraints take the form of natural language sentences, which may then be translated into model-based constraints and are used to identify boundaries on system behaviour such that hazardous states are not entered into.
Machine	A machine in Event-B represents part of a model and contains variables, invariants, theorems and events. A machine may refine another machine and 'sees' a context.
Context	A context in Event-B contains carrier sets, constants, axioms and theorems. A context can extend another.
Event	An event in Event-B represents a transition. Events contain guards (which constrain when they can occur), parameters, witnesses (if extending an abstract event) and actions. The transition represented by an Event must maintain all the invariants in the model to which they belong. Events can refine abstract events if the Machine they reside in is the refinement of another Machine.
Guard	A guard represents a condition under which the Event may occur. If all the guards of an event are true then the Event is permitted to occur.
Constants	Constants represent either sets or numeric constants. They then have their behaviours represented by axioms or theorems within the sets.
Sets	Sets represent the mathematical notion of sets and may have axioms and theorems apply to them.
Axioms	Axioms are behaviours or characteristics that sets or constants definitely have and are taken as correct.
Invariant	Invariants represent assertions about the properties of the model that must remain true.
Variants	Variants are utilised in models where events may have convergent or divergent behaviours.

verify overall system behaviour. This can be of significant aid when carrying out the modified STPA analysis described by this paper in an iterative fashion over multiple system designs and then down into component-level analysis.

3.3. Broad steps of the methodology

We will present each step of the methodology in its own paragraph to present the concrete actions required at each step. There may also be some small examples of artefacts from some of the steps in order to highlight the ideal outputs from those steps.

3.3.1. Step 1 - Establishing the system engineering basis. We begin by taking the system under analysis and defining its boundaries as well as the underlying purpose of the system. The underlying purpose and system boundaries should be explained in plain English, in the form of a statement. An example of such a statement would be:

> The purpose of this system is to maintain the temperature of a hydroponics facility such that the seedlings and plants do not expire due to overheating or freezing and equally that the work environment remains tolerable for those personnel working within the hydroponics areas of the facility.

The purpose statement is then used to identify potential losses - outcomes or system states where the system may fail to fulfil its purpose in any way. Following on from the example above, one potential loss state might be *'Plant and seedling matter expires due to poor temperature control'*, another may be *'Harm to personnel due to excessive temperature within facility'*. These loss states then allow us to define top-level system hazards. These hazards should attempt to encapsulate all ways in which the system may reach the loss state in combination with worst-case environmental conditions. One example of a worst-case environmental condition is a heatwave, which in our example could be combined with inadequate ventilation or cooling. This would result in the following hazard state: *'Temperature within seedling growth zone exceeds upper bound'*. Later steps should then identify much more specific hazards at the *component* level which are connected with the system-level hazards.

3.3.2. Step 2 - Build the control structure. The next step is to build a functional control structure diagram for the system under analysis. The route to building this depends on whether this analysis is being performed on a system that already exists or one that is still in the early design stage:

1) **For systems already in existence:** This is performed by identifying the main components of the system and determining which components interact, which components control which other components, what underlying processes are being managed/controlled, etc. This process may be aided by review of the existing documentation for the system. A brief overview of how this process is performed can be found in the STPA Primer [13].
2) **For systems that have not already been implemented:** If the system has open questions with regards to the architecture (i.e. a system that exists only in terms of functional/non-functional requirements), a proposed system design can be analysed to help aid in the process of finalising the system design. Systems with existing designs can equally be analysed as each pass of analysis should suggest refinements or changes to the design which can be integrated to increase the security/safety properties of the resulting system. In either case, elements of the control structure may not be explicit and may need to be inferred in order to entirely represent the system under analysis.

In either case, this functional control structure diagram should capture the processes and interactions between controllers and processes, as well as controllers and other controllers. Examples of functional control structures can be found in much of Leveson's work [11].

3.3.3. Step 3 - Generate control actions. The third step of the analysis is to generate a list of control actions - these should be derived from the control structure for the system from Step 2 wherever commands are exchanged between any elements. It may be appropriate to also identify the data types that are involved in each control action - with existing systems this should be clear while systems that are yet to be integrated can utilise requirements to derive this information.

3.3.4. Step 4 - Build the formal model. This step involves representing the control structure and control actions using the Event-B formal method. As there is ordinarily a fairly clear mapping from control actions to the notion of Events in Event-B, it is suggested that most control actions should be representable as events. The underlying state information maintained by the system will either be representable as variables within the Event-B Machine of the model of the system, or as constants in the Context of the system. Existing data typing should be expressed through axioms in the Context for constants, or through invariants on variables in the Machine. Guards within Events can then be utilised to ensure control actions occur in the right element of the system flow and that variables are of the correct type and status for the control action to occur.

3.3.5. Step 5 - Scenario analysis and critical requirement generation. The control actions are then subjected to analysis through considering if any insecure/unsafe system states (or *negative consequences* or *hazardous behaviours*) can occur if:

- The control action is issued.
- The control action isn't issued.
- The control action is issued too soon or too late within the ordinary sequence of system events.

- The control action is issued for too long/too short of a period of time.

If there is the potential for a control action to cause an unsafe/insecure system state as a result of one of these conditions, the *hazardous behavour* is noted down (a tabular form is often used for this in standard STPA - with the four conditions as columns, and each control action as a row). The hazardous behaviours of each control action can then be used to generate 'critical requirements' which represent constraints or limits in natural language on when/how control actions may be issued.

An illustrative example of a control action undergoing analysis following our previous example is the *'Turn on seedling heater'* control action which - when issued too soon or for too long - may result in a state in which seedling temperature is too high. The resultant 'critical requirement' in this circumstance would be *"Seedling zone heater may not be activated when temperature enters critical range"*.

3.3.6. Step 6 - Critical requirement integration. The existing set of critical requirements can then be translated into constraints and integrated into the formal model of the system.

This usually takes at least one of the following forms and may involve a combination:

- Refinement and data-refinement, through which the existing variables and events in the model are respectively refined through the creation of a descendant of the existing model.
- Addition of invariants to the Machine element of the model in order to constrain variables.
- Addition of guards to Events to narrow the circumstances in which they may occur.
- Addition of axioms to the Context element of the model to add properties to the constants and sets represented therein.
- Addition of new events, variables and other elements to the model and restricting existing variables, events, etc.

This should be undertaken in a refinement of the model for each critical requirement, where a refinement is possible. Each modification to the model should be recorded under which critical requirement it has mitigated against for traceability and assurance purposes.

Each refinement of the model can then be subjected to model checking as well as interactive proving to ensure that the model of the system does not deadlock, that the invariants and theorems introduced to the model cannot be contradicted through some series of events, and that the system's formal properties can still be verified.

The critical requirement example around the seedling zone heater can therefore be translated into a *Guard* on all *Events* which involve engaging the seedling zone heater to ensure that temperature is not within a pre-set temperature range. It could be further strengthened by placing an *Invariant* into the model which states that the temperature cannot be within the pre-set temperature **and** the heater engaged at the same time. This would cause the model to halt in the event that not all events correctly integrate the appropriate guards and cause this invariant to be false.

3.3.7. Step 7 - Causal factors analysis. At this stage, it is important (particularly when attempting to identify and mitigate against insecure system behaviours) to not only question when hazardous behaviours may emerge but also *how* they might emerge. This is primarily done by analysing the information channels across the entirety of the system to identify how information may be modified, replayed or intentionally jammed using the functional control diagram. By carrying out this analysis, it may be clear that in some cases that a failure in communication or an adversarial third party between elements in the control layer diagram may lead to unsafe/insecure system states. These hazardous behaviours or circumstances can then have critical requirements generated for them and these can equally be integrated into the model and verified for correctness.

Continuing our example, the heater may be erroneously engaged manually within the temperature cut-off range due to *inaccurate reporting of temperature* to the operator. This is a casual factor which may be due to component failure or malicious actors involved in the system control loop. This is not something likely to be highlighted by the initial pass of analysis, but is as important to mitigate against, especially from a security perspective.

3.3.8. Step 8 - Iteration and scoping. One key element of the STPA family of methodologies is their focus on not simply carrying out analysis as a one-off effort but as a continuous process to iterate the design to a state where as much of the hazardous behaviour is mitigated against as possible within the system design. It may be that the identification of the hazardous behaviours and the resultant critical requirements indicate that the underlying system architecture requires a significant review. In this circumstance, it would be appropriate to begin this process from the beginning as any design changes will likely modify the functional control structure or the control actions within the system under analysis, which will necessitate beginning again. This analysis can additionally be restarted from the beginning but be scoped in down to specific subsystems or portions of the functional control diagram to generate component-level hazards.

3.4. Comparisons to existing methodologies

We believe our modified STPA methodology to provide added value over the existing STPA-derived methodologies due to the inclusion of the Event-B formal method. The majority of STPA-derived methodologies in the literature view their generated constraints as the primary artefact and output of the process. These constraints are not inherently useful when considered in isolation as they will still need to be integrated into subsequent steps of the system engineering process. Furthermore, there is no verification that the

generated constraints fully mitigate against the hazardous behaviour that they are created to prevent.

The addition of the Event-B formal method addresses both of these issues to a significant degree. The formal verification of the critical requirements ensures (as discussed in the previous subsection) that the critical requirements cannot be bypassed by a series of events occurring, and also ensures that there is a formal proof associated with each invariant, guard and theorem added to the model. In terms of ensuring that the model remains relevant and useful throughout the remaining steps of the system engineering process, the model of the system can be utilised until the implementation stage of the system life-cycle due to the previously stated support for refinement and decomposition within the Event-B formal method. Furthermore, some work has been undertaken in producing test cases from a model represented in Event-B which may then be utilised to test controllers within the system [23].

4. Conclusion

In this paper, we presented a methodology to leverage an existing system analysis technique from which we derive system-level constraints in the form of 'critical requirements' when determining unacceptable system states and behaviours. We then translate these critical requirements into constraints on a formal model (represented in Event-B) which allows us to demonstrate that the constraints mitigate sufficiently against the identified hazardous behaviours and allow us access to tool support in the format of automated verification and model checking capabilities.

4.1. Future work

Future work will involve the creation and refinement of the associated tools which are used during the analysis portion of our methodology so as to reduce the workload of the analyst and provide the potential for information presentation beyond simple tables. An additional goal of future work will be to develop a corpus of commonly identified critical requirements and sample representations of these critical requirements within the model as to aid analysts in representing these critical requirements they may encounter.

References

[1] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Proceedings of the 47th Design Automation Conference*. ACM, 2010, pp. 731–736.

[2] R. Baheti and H. Gill, "Cyber-physical systems," *The Impact of Control Technology*.

[3] A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, and S. Sastry, "Challenges for securing cyber physical systems," in *Workshop on Future Directions in Cyber-physical Systems Security*. DHS, July 2009. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/601.html

[4] H. Cai, W. H. Wu, C. D. Zhang, T. K. Ho, and Z. M. Zhang, "Modelling safety monitors of safety-critical railway systems by formal methods," in *Railway Condition Monitoring (RCM 2014), 6th IET Conference on*, Sept 2014, pp. 1–5.

[5] R. Ross and J. C. OREN, "Systems security engineering, considerations for a multidisciplinary approach in the engineering of trustworthy secure systems," *NIST Special Publication*, vol. 800, p. 160.

[6] G. B. Dyson, *Darwin Among the Machines: The Evolution of Global Intelligence*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997.

[7] F. A. Manuele, *On the practice of safety*. John Wiley & Sons, 2013.

[8] F. E. Bird and R. G. Loftus, *Loss control management*. Institute Press, 1976.

[9] N. G. Leveson and M. Stringfellow, "Systems approach to accident analysis," Tech. Rep.

[10] J. Reason, "The contribution of latent human failures to the breakdown of complex systems," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 327, no. 1241, pp. 475–484, 1990.

[11] N. Leveson, *Engineering a safer world: Systems thinking applied to safety*, 2011.

[12] N. Leveson, N. Dulac, D. Zipkin, J. Cutcher-Gershenfeld, J. Carroll, and B. Barrett, "Engineering resilience into safety-critical systems," *Resilience Engineering–Concepts and Precepts. Ashgate Aldershot*, pp. 95–123, 2006.

[13] N. Leveson, "An STPA Primer - Version 1," Online, June 2015, http://psas.scripts.mit.edu/home/wp-content/uploads/2015/06/STPA-Primer-v1.pdf.

[14] W. Young and N. G. Leveson, "An integrated approach to safety and security based on systems theory," *Commun. ACM*, vol. 57, no. 2, pp. 31–35, Feb. 2014. [Online]. Available: http://doi.acm.org/10.1145/2556938

[15] S. S. Shapiro, "Privacy risk analysis based on system control structures: Adapting system-theoretic process analysis for privacy engineering," in *2016 IEEE Security and Privacy Workshops (SPW)*, May 2016, pp. 17–24.

[16] I. Friedberg, K. McLaughlin, P. Smith, D. Laverty, and S. Sezer, "STPA-SafeSec: Safety and security analysis for cyber-physical systems," *Journal of Information Security and Applications*, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214212616300850

[17] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald, "Formal methods: Practice and experience," *ACM Comput. Surv.*, vol. 41, no. 4, pp. 19:1–19:36, Oct. 2009. [Online]. Available: http://doi.acm.org/10.1145/1592434.1592436

[18] C. Newcombe, T. Rath, F. Zhang, B. Munteanu, M. Brooker, and M. Deardeuff, "How Amazon Web Services uses formal methods," *Commun. ACM*, vol. 58, no. 4, pp. 66–73, Mar. 2015. [Online]. Available: http://doi.acm.org/10.1145/2699417

[19] J.-R. Abrial, *Modeling in Event-B: system and software engineering*. Cambridge University Press, 2010.

[20] A. Rezazadeh, N. Evans, and M. Butler, "Redevelopment of an industrial case study using Event-B and Rodin," 2007.

[21] M. Butler and D. Yadav, "An incremental development of the Mondex system in Event-B," *Formal Aspects of Computing*, vol. 20, no. 1, pp. 61–77, 2008.

[22] J. Colley and M. Butler, "A formal, systematic approach to STPA using Event-B refinement and proof," February 2013. [Online]. Available: http://eprints.soton.ac.uk/352155/

[23] Q. A. Malik, J. Lilius, and L. Laibinis, "Scenario-based test case generation using event-b models," in *2009 First International Conference on Advances in System Testing and Validation Lifecycle*, Sept 2009, pp. 31–37.