

Highlights

An STPA-based Formal Composition Framework for Trustworthy Autonomous Maritime Systems

Dana Dghaym, Thai Son Hoang, Stephen Turnock, Michael Butler, Jon Downes, Ben Pritchard

- Propose a generic approach for requirements elicitation using formal methods.
- Apply an STPA-based hazard identification method integrated with formal modelling to generate both safety and security requirements.
- Suggest a compositional approach for formal modelling to enhance the hazard analysis process and address challenges in formal modelling.
- Apply our approach of requirements generation and the SE-STPA hazard analysis approach to an integrated mission management system of autonomous vehicles.

An STPA-based Formal Composition Framework for Trustworthy Autonomous Maritime Systems[★]

Dana Dghaym^{a,*}, Thai Son Hoang^a, Stephen Turnock^b, Michael Butler^a, Jon Downes^b and Ben Pritchard^c

^a*School of Electronics and Computer Science, University of Southampton, Southampton, UK*

^b*Maritime Robotics Lab, University of Southampton, Southampton, UK*

^c*Thales UK*

ARTICLE INFO

Keywords:

Formal Methods; STPA; Event-B; Requirements; Maritime Autonomous Systems; SE-STPA

ABSTRACT

A key risk with autonomous systems (AS) is the trustworthiness of the decision-making and control mechanisms that replace human control. To be trustworthy, systems need to remain safe while being resilient to unpredictable changes, functional/operational failures and cybersecurity threats. Rigorous validation and verification are essential to ensure trustworthiness of AS. Current engineering practice relies heavily on Verification and Validation (V&V) test-and-fix of system characteristics which is very time-consuming and expensive, limiting the possibilities for exploration of alternatives in system design. Instead, we focus on building a correct-by-construction system. In this paper, we present an approach to identifying and analysing mission requirements for squads of autonomous missions. Clear definition of requirements is an important pre-requisite for mission planning and for V&V of mission management. We use a structured approach to requirements identification and use formal modelling to help remove ambiguities in the requirements and to specify formal properties that should be satisfied by the missions. Our approach uses a combination of analysis techniques based on *Systems Theoretic Process Analysis* (STPA) and formal modelling to generate critical requirements that ensure the safety and security of the system. We also suggest a compositional approach for formal modelling to enhance re-usability and address the complexity of formal modelling. Our approach is being evaluated through consideration of a combined mission of an Unmanned Surface Vehicle (USV) with deployment/recovery of small Unmanned Underwater Vehicles (UUV) within a shipping channel whereby the USV has to safely maintain station for a long period and then proceed to recover the UUV, while maintaining a communication link to an Unmanned Aerial Vehicle (UAV).

1. Introduction

Autonomous Systems (AS) offer the potential of reducing the cost and ensuring the safety of humans. However, managing a squad of heterogeneous AS can be costly requiring a large number of people to complete a mission. This paper will focus on the early phases of designing an integrated mission management system for heterogeneous autonomous assets, which we call the *Integrated Mission Management System* (IMMS). The aim of this system is to reduce the cost of missions requiring multiple platforms.

Mission management involves the following activities: planning of a mission after identifying the mission goals, mission execution, and mission monitoring and reviewing. While we have trustworthy autonomous vehicles working as separate entities, our aim is to build a trustworthy management system to ensure the trustworthiness of the overall system.

Studies have shown that the cost of fixing errors during testing is 10 times more than during the construction phase


and can increase to more than 25 times post release (Leffingwell, 1997), and many problems discovered in software systems are related to shortcomings in requirements elicitation and specification processes (MacDonell et al., 2014). In this paper, we are interested in maritime missions with autonomous vessels for each domain, aerial, surface and underwater. We will show how we can apply formal modelling to develop a requirements analysis framework for identification of anticipated range of operational environments for autonomous missions, including human operator interactions, together with precise specification of safety and security envelopes for enactment of autonomous missions.

The IMMS involves several components that interact with each other adding more complexity to the system. Several studies such as Bensaci et al. (2018) and Zhou et al. (2019) have shown the advantage of *Systems Theoretic Process Analysis* (STPA) over traditional hazard analysis techniques when considering the interaction between different components. While STPA lacks the formal aspect, combining it with the formal method, such as Event-B, adds more assurance by proving the consistency of the requirements. Event-B is a system analysis and modelling language that has a good extensible tool support which combines both theorem proving and model checking. Therefore, our choice of combining both Event-B and STPA can add more reassurance for a complex system such as the IMMS.

Our contributions can be summarised as follows:

[★]This is funded by Thales IMMS 2019 project.

*Corresponding author

 d.dghaym@ecs.soton.ac.uk (D. Dghaym);

t.s.hoang@ecs.soton.ac.uk (T.S. Hoang); s.r.turnock@soton.ac.uk (S. Turnock); mjb@ecs.soton.ac.uk (M. Butler); j.j.downes@soton.ac.uk (J. Downes); ben.pritchard@uk.thalesgroup.com (B. Pritchard)

ORCID(s): 0000-0002-2196-2749 (D. Dghaym); 0000-0003-4095-0732 (T.S. Hoang); 0000-0001-6288-0400 (S. Turnock); 0000-0003-4642-5373 (M. Butler); 0000-0003-2027-4474 (J. Downes)

- Define an approach for requirements elicitation based on formal modelling, that is generic and can be applied to any integrated system. We propose this approach in (Dghaym et al., 2019), and here we extend it to derive additional safety and security requirements as below.
- Apply an approach based on Systems Theoretic Process Analysis (STPA) to analyse the security requirements, and also to generate any additional safety requirements.
- Propose an improvement to Systems Theoretic Process Analysis (SE-STPA) formal modelling, that simplifies formal modelling and supports re-usability and change without breaking correctness.

This paper is organised as follows: Section 2 presents some related work. In Section 3 we present the case study that motivated our work. Section 4 gives an overview of our approach for identifying and analysing the mission requirements. Section 5 introduces the Event-B formal method and shows how we apply formal modelling using Event-B to derive the requirements. Section 6 demonstrates the SE-STPA hazard analysis and describes how we apply SE-STPA to do the hazard analysis of the IMMS, while in Section 7 we focus on the adversary modelling. Finally, we conclude in Section 8 discussing the advantages and limitations of our approach.

2. Related Work

STPA (Leveson and Thomas, 2018) is a hazard identification method which represents the system by a control structure. The hazardous conditions are generated by the absence, presence or the improper timing of control actions. The process is followed by identifying causal factors for unsafe control actions. On the other hand, formal methods are concerned with ensuring the system's reliability by specifying and verifying the correct behaviour of the system. Approaches that combines analysis techniques such as STPA with formal methods provide stronger arguments to trust the system's correct and safe behaviour.

Thomas and Leveson (2013) have defined a formal syntax for hazardous control actions, which are identified as a result of applying STPA. This formalisation enables the automatic generation of model-based requirements as well as detecting inconsistencies in requirements.

Abdulkhaleq et al. (2015) propose a safety engineering approach that uses STPA to derive the safety requirements and formal verification to ensure the software satisfies the STPA safety requirements. These safety requirements can be formalised and expressed using temporal language. For verification, a behaviour model corresponding to the controller's behaviour and constrained by the STPA requirements is constructed. From the behaviour model, an input model can be manually constructed and fed to a model checker such as NuSMV for verification against the formalised STPA requirements. Then apply model-based testing to automatically generate test cases.

Colley and Butler (2013) present an approach to hazard analysis, where requirements are first captured and classified as: monitored, controlled, mode and commanded phenomena. Then STPA is applied to the controlled phenomena which results in additional safety constraints. The derived safety constraints are then formally modelled using Event-B as invariants and/or guards.

Hata et al. (2015) also combine STPA with formal modelling. They apply STPA to derive the critical constraints which are formally modelled as pre and post conditions in VDM++.

While STPA is used for safety problems, some approaches build on STPA to introduce security analysis, such as STPA-Sec (Young and Leveson, 2014) which is an extension to STPA. STPA-Sec applies similar steps to STPA for security analysis, but differs in the addition of intentional actions when identifying causal scenarios.

Friedberg et al. (2017) present an integrated analysis technique based on STPA called STPA-SafeSec. This framework unifies the analysis of both safety and security and shows the dependency between the two domains. STPA-SafeSec introduces a component layer that identifies the physical impact of security vulnerabilities on the system.

Omitola et al. (2019) apply STPA for the security analysis and security requirements elicitation of a maritime communication system. They apply the same STPA safety concepts to security analysis, where they identify losses, security constraints and insecure actions instead of the STPA hazards, safety constraints and unsafe control actions.

Our approach uses both an STPA based approach (SE-STPA) and a formal method (Event-B) to generate the system's critical requirements which includes both safety and security constraints. Our approach differs by starting with formal modelling of the high level system which also helps to extract and improve the functional requirements. Our approach is iterative combining the analysis techniques with formal modelling. However, formal modelling can get complex with the continuous addition of newly generated critical requirements. We show, using a complex case study, how we apply different techniques based on decomposition and composition to simplify the formal modelling process and enhance the re-usability of the models even when changes are required.

3. Case Study

The purpose of the IMMS is to reduce the large number of people running separate platforms to a more integrated system requiring less people. The use case scenario that motivated building the IMMS is represented in Figure 1.

The overall goal of the use case is to conduct a timely survey of the seabed in a defined area. The unmanned underwater vehicles (UUVs) conduct the seabed survey which is sent to the host ship via the unmanned surface vehicle (USV). The unmanned aerial vehicle (UAV) can possibly play a gateway role between the USV and the host ship, but in our experiments the UAV was mainly providing situational awareness.

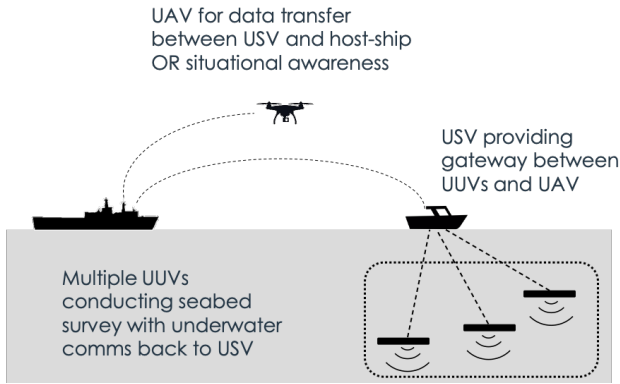


Figure 1: IMMS Use Case Scenario

The IMMS role is to provide the high level plans for all the different platforms which are sent via the *Ground Control Station* (GCS) to the different assets, which in turn send updates and data to the IMMS through their corresponding GCS.

4. An Approach for Requirements Elicitation

Defining the requirements of the IMMS is an iterative process, where in the initial version we focus on **what we know**. We start by gathering information about the different available autonomous platforms. In this case, we have three different physical assets from each of the three domains: surface, underwater and aerial. After defining the system goals, assumptions and constraints, which include communication and planning constraints in addition to identifying the failure and adverse conditions, we structure the requirements as follows:

1. Operator Safety Requirements
2. Platform Functional Requirements
 - (a) Unmanned Surface Vehicle (USV)
 - (b) Unmanned Aerial Vehicle (UAV)
 - (c) Unmanned Underwater Vehicle (UUV)
3. Possible Exceptions and Recovery Actions
4. Security Requirements

In the initial version, our focus is on the available assets and their interfaces to the IMMS. After analysing the existing requirements, we identify what is missing, in this case it is clearly the IMMS requirements or in other words **what we want** to achieve from the IMMS. From what we know and what we want, we identify the functional and non-functional requirements of the IMMS, the IMMS interface requirements and information communication. The functional requirements include mission planning, mission execution and mission monitoring and review. Later, we can identify a common functionality among the different assets and generalise the platform-specific requirements.

Capturing the requirements in a well-defined natural language document is not enough. The document can be still prone to different interpretations from the different team members coming from different backgrounds. Therefore, it is important to have a precise specification to eliminate any ambiguities and remove any defects. For this we use Event-B, introduced in Section 5.1, to capture the system requirements precisely. In Section 5.2 we present an early attempt at modelling the high level requirements of the system using Event-B.

Figure 2, which we introduce in (Dghaym et al., 2019), presents our proposed approach for eliciting requirements for autonomous missions. This approach is based on our experience in using formal modelling for system verification, it is a generic approach which is applicable for the integration of multi-platforms. Our approach is iterative where we augment these requirements as a result of continuous analysis. This approach requires continuous interaction between two main stakeholders the domain experts, which in our case includes two parties: the different platform owners and the client, and the formal methods experts. The platform owners will identify what are the feasible requirements and the client identifies what is the purpose of the system. The formal methods experts will work on analysing the available resources to identify what is missing and what is ambiguous which will need clarification from the domain experts, who should also approve or reject any identified requirements.

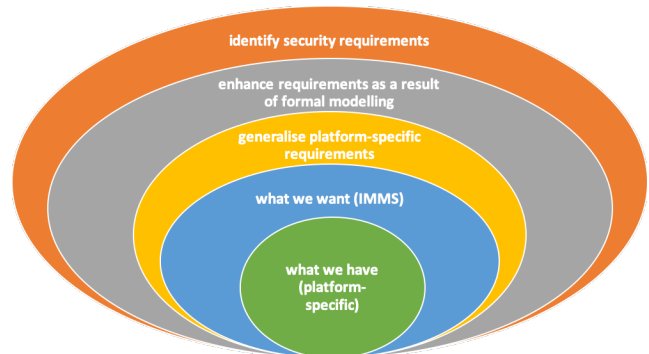


Figure 2: An approach for eliciting requirements

While Figure 2 focuses on the early phases of requirements elicitation, we augment this approach with a well defined process based on both formal methods and different analysis techniques as illustrated in Figure 3. The process starts with initial analysis of the system. The initial analysis includes the three internal circles of the requirements elicitation approach (Figure 2). The initial analysis will result in an initial system requirements which include both functional and non-functional requirements. The next steps will cover in more details the two external circles of Figure 2. We, then use the initially defined requirements to formally model the overall system which will generate additional requirements. Later we apply SE-STPA, where the first steps of SE-STPA will result in identifying the control actions, which are used to enhance our formal modelling. Then, we apply hazard

analysis of the control actions which results in the generation of additional critical requirements, which we formally model to ensure their consistency. Later, we apply the SE-STPA adversarial modelling which generates additional security-related critical requirements, which we also formally model. We then, do further causal analysis and analyse the results of our formal modelling to re-scope the system and repeat the process until we have a well-formed system requirements.

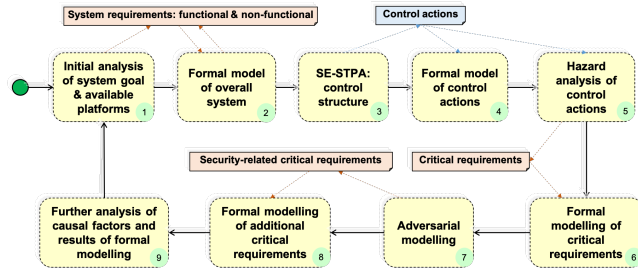


Figure 3: A process for requirements elicitation using analysis and formal methods

In the next sections, we will explain in more details the different steps of our process presented in Figure 3. We will also apply the process to generate the requirements of the IMMS, starting with the initial analysis as shown in Figure 2.

4.1. Analysis of an IMMS Safety Requirement

In this section, we will focus on one of the safety requirements of the IMMS, which we use as a running example to illustrate our approach rather than presenting the full set of requirements. In the first stage we looked at identifying the operator Safety Requirements (SF) of the available assets, one of these requirements is:

SF1. Collision Avoidance (CA): The UxVs (unspecified unmanned vehicles) do not have collision avoidance mechanisms. Collision avoidance with other vehicles and other possible obstacles is maintained by thorough planning. The plans sent to the assets can be updated during the mission should conditions change, for example the appearance of new obstacles. Additionally, maintaining *visual line of sight*, receiving video feeds from platforms and defining mitigation scenarios assist with collision avoidance.

CA Requirement Analysis: The available vehicles do not have collision avoidance mechanism. Therefore, in order to avoid collisions:

- A Initial planning should take into considerations the different assets positions and any known obstacles in the environment.
- B During the mission when situational awareness is available and the assets are communicating, the plans can be updated and sent to the assets to avoid collisions.
- C A timeout should be predetermined for the assets with a predetermined plan to follow in case of communication loss.

Identifying IMMS functional requirements: By analysing SF1, we can identify some of the IMMS Functional Requirements (FR) which should include:

- FR1. The IMMS must have the ability to specify/assign the required vehicles to perform a mission.
- FR2. The IMMS must assign tasks to the specified vehicles.
- FR3. The IMMS must provide the vehicles with initial plans prior to starting a mission.
- FR4. The IMMS must have the ability to modify plans of assigned vehicles during the mission executions.

Both FR1 and FR2 can be inferred from A, since planning should know the positions of the mission assets, then it should have the ability to assign these assets to a mission and give them tasks to perform a mission. From both A and C, FR3 is deduced which will result in providing the vehicles with plans in the case of normal and failure behaviours. FR4 is clearly concluded from B where plans should be updated should problems arise.

In this section, we have shown how we can identify some of the system requirements by analysing the requirements of *what we know*. This is part of the initial analysis or step 1 of our process (Figure 3), which results in the identification of some functional and non-functional requirements.

5. Modelling in Event-B

In this section, we present Event-B (Abrial, 2010), which is a formal method for system analysis and modelling. We have chosen Event-B because it supports modelling at a system level rather than only at a software level. Event-B also has good extensible tool support and a user can apply both theorem proving and model checking, supported by ProB (Leuschel and Butler, 2008), to the same model. A survey of formal verification tools have found that Event-B supported by the toolset Rodin comes closest to supporting the goals of a correct-by-construction designs (Punnoose et al., 2014). In this paper, we use Event-B to address the ambiguity and inaccuracy of requirements specifications. Event-B, like many other formal methods, uses concise mathematical language for specification. Moreover, safety properties expressed as invariants are verified which helps to eliminate any inconsistency in the model. Formal models written in Event-B can also be animated to validate against user requirements.

5.1. Event-B

Event-B (Abrial, 2010) is a formal method for system development. One of the main features of Event-B is the use of refinement to introduce system details gradually into the formal model. An Event-B model consists of two parts: contexts and machines. Contexts are the static parts of the model. A context contains carrier sets, constants, and axioms that constrain the carrier sets and constants. Machines are the dynamic parts of the model. A machine contains variables v , invariants $I(v)$ that constrain the variables, and

events. An event comprises a guard denoting its enabling-condition and an action describing how the variables are modified when the event is executed. In general, an event e has the following form, where \mathbf{t} are the event parameters, $G(\mathbf{t}, \mathbf{v})$ is the guard of the event, and $\mathbf{v} := E(\mathbf{t}, \mathbf{v})$ is the action of the event.

event e == **any** \mathbf{t} **where** $G(\mathbf{t}, \mathbf{v})$ **then** $\mathbf{v} := E(\mathbf{t}, \mathbf{v})$ **end**

Event e can only be executed when its guard $G(\mathbf{t}, \mathbf{v})$ holds for some parameter \mathbf{t} and its effect on the state variable \mathbf{v} is specified by the action $E(\mathbf{t}, \mathbf{v})$.

A machine in Event-B corresponds to a transition system where variables represent the states and events specify the transitions. Contexts can be extended by adding new carrier sets, constants, axioms, and theorems. Machine M can be refined by machine N (we call M the abstract machine and N the concrete machine). The state of M and N are related by a gluing invariant $J(\mathbf{v}, \mathbf{w})$ where \mathbf{v}, \mathbf{w} are variables of M and N , respectively. Intuitively, any “behaviour” exhibited by N can be simulated by M , with respect to the gluing invariant J . Refinement in Event-B is reasoned event-wise. Consider an abstract event e and the corresponding concrete event f . Somewhat simplifying, we say that e is refined by f if f ’s guard is stronger than that of e and f ’s action can be simulated by e ’s action, taking into account the gluing invariant J . More information about Event-B can be found in (Hoang, 2013) Event-B is supported by *Rodin platform* (Rodin) (Abrial et al., 2010), an extensible toolkit which includes facilities for modelling, verifying the consistency of models using theorem proving and model checking techniques, and validating models with simulation-based approaches.

In Hoang et al. (2017), we introduce a mechanism for machine inclusion in Event-B. Machine inclusion allows the construction of an Event-B machine by composing one or more machines and synchronising events. The included machine is reused in a *correct-by-construction* fashion that allows to utilise its properties without the need of re-proving. This compositional approach makes it possible to combine both top-down (supported by refinement) and bottom-up development. A machine M including machine M' , will inherit the variables and invariants of M' , and will be able to synchronise with any of its events. If e in M synchronises with e' in M' , then e will inherit the parameters, witnesses, guards and actions of e' .

5.2. Formal Modelling of IMMS

A key strength of Event-B is refinement, which allows us to abstract away from details and focus on different problems at different levels of refinements. The goal of this early modelling is an attempt to understand the system under development, remove ambiguities and identify important missing properties of the system to enhance the requirements.

Our Event-B model starts with an abstract level defining a mission as a set of tasks. Then, we introduce two refined levels as: vehicles and mission planning.

Abstract Level: At the abstract level we define a mission as a set of tasks and introduce the following events executing in the corresponding order:

`<define_mission; start_mission; complete_mission >.`

This level has three simple events, however right at the start of modelling we have to take a modelling decision: Can the IMMS manage multiple simultaneous missions?

For this project we will manage one mission at a time and leave this question as a future research question. Other questions that we have identified at this level are:

- What are the conditions for starting a mission, or we can ask, do we need all the vehicles to be present to start a mission?
- What are the conditions for completing a mission?

We can define a new requirement for the IMMS, related to starting a mission:

FR5. The IMMS should define a minimum criterion for starting a mission.

First Refinement: In the first refinement level, we introduce vehicles and their capabilities and the possibility of assigning vehicles to mission tasks. The main functionality at this level is to ensure that a mission can only start after assigning vehicles with the minimum required capabilities defined to start the mission tasks. This also addresses FR5. In Event-B this can be modelled by the following invariant which must be maintained by all the events.

```

1 missionStart = TRUE ⇒ (
2   requiresMin[missionTasks]
3   ⊆
4   capabilities[assign-1[missionTasks]]
5 )
```

Here $\text{assign}^{-1}[\text{missionTasks}]$ denotes the set of vehicles associated with all mission tasks.

Proof obligations are generated to ensure that all events will maintain the defined invariants. However, if we allow assets to be deallocated from their tasks during the mission. The invariant for FR5 cannot be maintained, hence in this case it is better to only model it as a pre-condition of starting a mission in the form of a guard in `start_mission` event.

In addition to that Event-B can help to prove the consistency of the invariants, for example if we have additional invariants that conflict with each other, it will be impossible to prove the model and hence it will flag a problem to the modeller and requirements can be changed accordingly.

Second Refinement: At the second refinement, we introduce mission plans abstractly as a series of locations, and in our model we ensure that a mission is not considered successfully complete until all vehicle plans are covered. We also introduce an event to set an initial plan before starting a mission and another event that enable modifying plans during execution, addressing requirements FR3 and FR4.

This level also poses new questions about the conditions for modifying plans, is it always a response to some changes

to the environment, do we immediately modify the plan or does the vehicle has to go through a safe state?

To answer these questions, we suggest defining generic states that apply to all the vehicles and define “what are the activities that can occur during these states?”. These activities and the state transitions will be modelled as events in Event-B. Later, we can refine the generic states by defining the tasks specific to each vehicle.

In this section, we have shown an example of how we can use Event-B to improve the requirements and identify some defects. More details about the initial system model can be found in (Dghaym et al., 2019). This section is part of step 2 of Figure 3, where we have used the initial requirements in the formal model of the high level system and generated additional requirements as the result of formal modelling.

6. An STPA-Based Approach

In this section, we introduce SE-STPA (Howard et al., 2019), which is an STPA-based approach for the safety and security analysis of systems. We chose SE-STPA because it integrates formal modelling to the analysis process to provide more assurance to the analysis process which involves both safety and security.

We show how we apply the SE-STPA approach, to help us identify the safety and security requirements, which also covers the last circle of our proposed approach for requirements elicitation, Figure 2. We also propose a modification to the formal modelling strategy, where the original SE-STPA relies only on refinement. Our formal modelling is based on decomposition, composition and separation of concerns in order to tackle the complexity in formal modelling and verification of the critical requirements and support reusability.

6.1. SE-STPA

Security-Enhanced Systems-Theoretic Process Analysis (SE-STPA) is an STPA-based methodology developed by Howard et al. (2019). This methodology integrates the STPA analysis with formal modelling using Event-B. SE-STPA does not only consider safety analysis, but also adds the adversarial modelling of the security risks.

SE-STPA is an iterative methodology which consists of the following eight steps:

1. **Establishing the system engineering basis:** this involves identifying the system entities and boundaries from the system purpose. As well as, identifying the system losses and hazards.
2. **Building the functional control structure:** this is a representation of all the entities involved in the control of the system and any underlying processes.
3. **Identifying control actions:** these can be identified from the commands and feedback in the control loop of the functional control structure.

4. **Building the initial formal model:** from the control actions, an abstract Event-B model is built where control actions represent events in the Event-B machine.
5. **Hazard analysis of the control and critical requirements identification:** the control actions are analysed to identify any hazards, this is done by checking if the control action is issued, not issued, issued out of sequence or issued for the incorrect duration can lead to any hazardous consequence. Critical requirements can then be identified from the hazard analysis.
6. **Adversarial modelling and further generation of critical requirements:** an adversary is an abstraction of any unauthorised party that may undermine the purpose of the system. By analysing the possible manipulation points and actions that an adversary may undertake, further requirements can then be generated to secure the system. The purpose of adversary modelling is to provide a security assurance case.
7. **Integration of the generated critical requirements:** the generated critical requirements can be integrated into the initial formal model through different refinements.
8. **Causal factors analysis:** this is related to the hazards identified from the control actions rather than the adversary modelling. Hazards are analysed to identify what series of actions or context can lead to the hazards. Possible additional requirements can also be generated.

Next, we present our application of SE-STPA to IMMS, which covers steps 3 and onwards of our process in Figure 3.

6.2. Engineering Basis & Control Structure

From the initial requirements analysis presented in Sections 4 and 5, we can summarise the purpose of the IMMS as follows.

The IMMS is an overarching architecture framework that allows the interoperability of diverse sets of AS. At its heart the IMMS is a software which manages the interfaces and exchanges between these different autonomous assets. The IMMS will be able to perform a global Tier 1 planning and provide a mission definition across multiple platforms. IMMS will also be able to communicate with the different assets and monitor the mission during its execution and update plans if needed.

From the purpose statement, we can see that communication with the different assets plays an important role especially from a security perspective. The IMMS can communicate with the different assets via their GCS. Therefore, the system has four entities:

- **Operator:** Set mission goals and set restrictions (no-go zones) these could be imported or set manually by the operator.

- **IMMS Mission Planner Software (IMMS):** Generate plans based on mission goals, these plans should abide by the restrictions, i.e., avoid the no-go zones. The IMMS should be able to communicate with the ground control stations to send the mission plans across the different domains.
- **Ground Control Stations (GCS):** Interface between the IMMS and the assets, transmitting Tier 2 plans to the registered assets.
- **Unmanned Vehicles or Assets (UxV):** A number of unmanned vehicles from the different domains that fulfils the planned mission routes.

Next, identify the system losses and hazards and the mapping between them. Table 1 presents an example of a loss in the system and its associated hazards.

Table 1
Losses and hazards mapping

Loss	Hazard
L1. GCS and/or IMMS are not certain of the location of the UxV.	H1. UxV deviates from the plan. H2. UxV do not report position and telemetry data regularly during mission.

6.2.1. Control Structure

The four entities identified in the previous step represents the components of the functional control structure in Figure 4. The control structure presents the communication between the different components, where commands and feedback can be exchanged between the different components. In the control structure representation (Figure 4), we identify the responsibilities of each controller and its process model which represents the aspects of the system relevant to the controller and its responsibilities. As can be seen in the control structure of the IMMS the plans and commands are passed from the IMMS to the assets via the GCS. On the other hand, location and state updates initiate from the assets to the IMMS via the GCS. The operator can directly view updates received by the IMMS and define/update mission goals to the IMMS.

The control actions can be identified from Figure 4. For example the IMMS control actions are: generate plans, validate plans while the operator authorise plans.

At this stage we have completed step 3 of Figure 3, where the output of this step is a set of control actions of the different entities of the system.

6.3. Formal Modelling of Control Actions

Following the SE-STPA approach, after identifying all the control actions the initial formal modelling is introduced. However, in our case we have started the initial formal modelling right from the beginning and we used these formal

models to introduce new requirements and improve our system understanding. At the beginning, our focus was at the system level and deriving the functional requirements and not the communications between the different entities. That is why in our initial models, we do not have the GCS as an explicit separate entity. For example, we do not distinguish between the plans the IMMS generates, the plans the GCS receives and the ones executed by the assets. To summarise, in the initial model we trusted the system communications and hence we assumed all the entities have the same information. However, this is not the case when analysing the safety and security hazards, where communications can introduce major threats that can lead to unsafe scenarios.

Unlike the SE-STPA defined by Howard et al. (2019), we have not used one abstract model, we have developed three abstract models to represent the three levels of communications shown in Figure 4. These three abstract models can then be composed into one model using inclusion (Hoang et al., 2017). This decomposition approach does not only simplify the modelling, but also emphasises the fact that the communicated messages are not necessarily the same and in the next sections we will focus on ensuring a secure and safe communications. In this model the communicated messages are mainly the plans sent from the IMMS to the assets via the GCS and the locations sent in the opposite direction from the assets to the IMMS via the GCS.

In Figure 5, we present the events (representing the control actions) of the three abstract models. These models are then combined into one model using machine inclusion and event synchronisation. The composed model is the formal model of the control actions represented in step 4 of Figure 3.

6.4. Hazard Analysis

Control actions are analysed by checking the consequence of issuing or not issuing them. As well as issuing the control action in the wrong sequence or for the wrong duration. This is slightly different to the traditional STPA, it is mainly a change in terminology to apply to both safety and security as both aspects are treated as first class citizens in SE-STPA. Table 2 presents the hazards from the IMMS control action “generate plan”. Issued for the wrong duration is not applicable for this control action because it is discrete.

After identifying the hazards in scope, critical requirements are generated. Table 3 shows an example of the critical requirements generated from the hazards in Table 2. This is step 5 of Figure 3.

For example, the control action “generate plan” of the IMMS if issued can have the potential hazard of generating plans containing no-go zones or hazardous locations H2. To mitigate for this hazard we generate the safety requirement CR2 which requires validating the generated plans against no-go zones before sending them to the assets.

6.5. Formal Modelling of Critical Requirements

Incorporating critical requirements in the formal model is an important step for verifying the system. If we take crit-

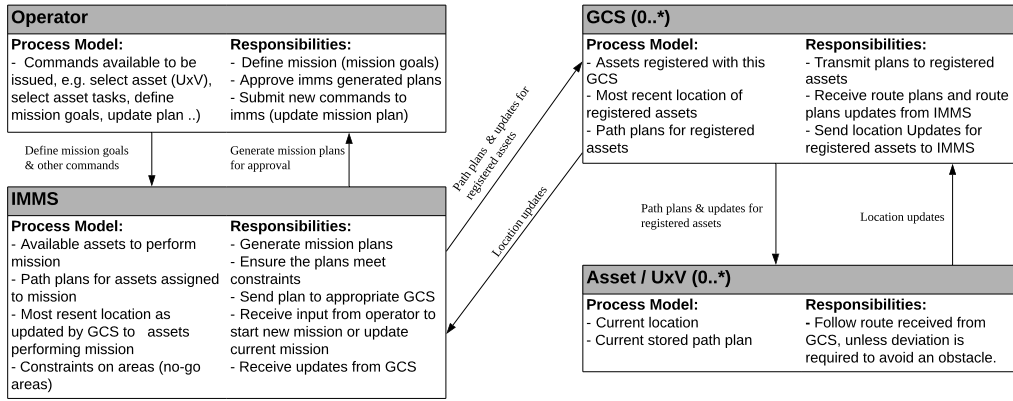


Figure 4: Functional control structure of the IMMS

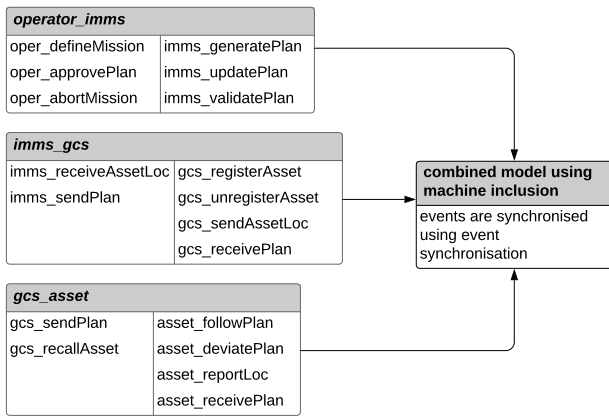


Figure 5: Abstract models events

Control Action	Is issued	Is not issued	Is issued out of sequence
Generate plan	Generated plans contain way points in no-go zones or hazardous locations.	Mission cannot be conducted if plans are not generated.	Start generating plans while mission goals are still being defined can leave the system in an indeterminate state.

Table 2
Hazard analysis of a control action

ical requirement CR2 in Table 3, this requirement is concerned with validating the generated plans to ensure the safety properties. One criteria for validating the plans is avoiding no-go zones.

Therefore, we develop a runtime policing function to validate the output of the mission planning system against the safety properties, in this case avoiding no-go zones. The

Table 3
Critical requirements generation from hazards

Hazard	Critical Requirement
H2. generated plans contain way points in no-go zones or hazardous locations.	CR2. Plans containing no-go zones or hazardous locations shall not be validated and sent to assets.
H3. Mission cannot be conducted if plans are not generated.	CR3. A mission can only start if plans are generated and validated.
H4. Generating plans while mission goals are still being defined can leave the system in an indeterminate state.	CR4. Plan generation can only start after all mission goals are confirmed.

policing function is independent of the of the system’s planning function. The decoupling of validation and planning allows us to update and replace the intelligent planning function, using the same validation approach. This approach for ensuring the safety of the IMMS is based on (Bogdiukiewicz et al., 2017).

Our development is based on Event-B refinement to link the abstract specification with the implementation. Our Event-B development can be summarised as follows:

1. Policing a collection of plans against all no-go zones.
2. Policing an individual plan against all no-go zones.
3. Policing an individual plan against a no-go zone.

In order to model the policing function for a collection of plans against all no-go zones, the Event-B context is defined as follows:

```

1 context c0_0
2 sets
3 AREA
4 PLAN
5 VVPROBLEM
6
7 constants
    
```

```

8 areas
9 individual_plans
10 nogo_problems
11 axioms
12 @typeof-nogo_zones: areas ⊆ AREA
13 @typeof-plans: individual_plans ⊆ PLAN
14 @typeof-conflict: nogo_problems ∈ PLAN × AREA ↔
    VVPROBLEM
15 end
    
```

The NOGO problems are abstractly specified as a binary relation between a tuple (plan, area) and some V&V problems. We will specify exactly what are the V&V problems later. To implement the gathering of NOGO problems, we iterate through the collection of individual plans and collect the problems gradually. This is represented in Machine `m0_1` below.

```

1 machine m0_1
2 refines m0_0
3 sees c0_0
4
5 variables
6 verified_plans
7 avoid_nogo_result
8 invariants
9 @typeof-plans_to_verify: verified_plans ⊆ individual_plans
10 @verify_result: avoid_nogo_result = nogo_problems[
    verified_plans × areas]
11
12 events
13 event INITIALISATION
14 then
15 @init-avoid_nogo_result: avoid_nogo_result := ∅
16 @init-verified_plans: verified_plans := ∅
17 end
18
19 event verify_avoid_nogo_individual
20 any individual_plan individual_result where
21 @grd1: individual_plan ∈ individual_plans
22 @grd2: individual_plan ∉ verified_plans
23 @grd3: individual_result = nogo_problems[{individual_plan} ×
    areas]
24 then
25 @act1: verified_plans := verified_plans ∪ {individual_plan}
26 @act2: avoid_nogo_result := avoid_nogo_result ∪
    individual_result
27 end
28
29 event verify_avoid_nogo
30 refines verify_avoid_nogo
31 where
32 @grd1: verified_plans = individual_plans
33 with
34 @result: result = avoid_nogo_result
35 end
36 end
    
```

Variable `verified_plans` stores the set of `verified_plans` so far and helps us model the iteration. As a result, this is always a subset of the input individual plans (invariant `@typeof-plans_to_verify`). The result is collected gradually in variable `avoid_nogo_result`. Invariant `@verify_result` specifies the consistency of the collected result: it must always be the NOGO problems restricted to the set of plans that have been verified so far.

Event `verify_avoid_nogo_individual` specifies the step

for verifying an individual plan. The individually verified result is stored in the event parameter `individual_result` and specified using guard `@grd3`. The set of verified plans is enlarged accordingly (`@act1`) and the individually verified result is merged with the collected result (`@act2`).

In later refinements we specify what are the `nogo_problems` which we specify as the intersection of the plans with the nogo areas. Plans are modelled more specifically as a series of waypoints. We do not show the rest of the Event-B development due to space limitation.

Here, we have shown how we validated the plans against the no-go problems, but we can also extend the model to include other safety problems.

Critical requirements `CR3` and `CR4` can be ensured in the overall system model introduced in Section 5.2. These requirements are concerned with the order of the events which can be ensured by event guards in Event-B. For example, we need to add a guard to `start_mission` from Section 5.2 to ensure all generated plans are valid, this is to satisfy `CR3`. However, proving the validity of plans is done in a separate model as described by the policing function. This separation simplifies the modelling and verification process.

In this section, we have shown how we can integrate the generated critical requirements to the formal model. We have used the safety requirement `CR2` as an example and applied a policing function to ensure the plans are valid. This approach can be easily extended with other safety requirements, for now we have used “avoiding no-go zones”. We have also shown how we can incorporate this requirement abstractly as part of the high level system model with minimal need to changes to the formal model. Our approach is different from the original SE-STPA approach (Howard et al., 2019), where each critical requirement can be added in a new refinement, which can be complex and require lots of changes and verification efforts.

7. Adversarial Modelling

As part of steps 7 and 8 of Figure 3, we need to do adversary modelling which should further generate security related critical requirements. These generated requirements will in turn need to be formally modelled. Adversary modelling starts by annotating the functional structure diagram with manipulation points. These manipulation points are the links between the different controllers or the controller and its processes, where an adversary can possibly have access to the system. Figure 6 illustrates the possible manipulation points of the IMMS framework.

There are different types of adversary with different severity levels. Adversaries can range from nation-state actors, hacktivist groups and organisations to curious individuals and unintentional adversaries. In this paper we will focus on manipulation point `MP2`, representing the communication between the IMMS and GCS.

Actions of an adversary are treated similar to hazards, because we do not have control over these actions. Critical requirements can then be generated similar to Section 6.4.

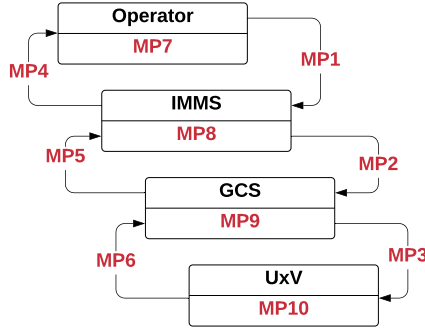


Figure 6: Manipulation points in the control structure

In Table 4, we demonstrate some of the adversary actions that can be carried over MP2 by different types of adversaries.

These security vulnerabilities, can have an impact on the system's safety. For example, H7 in Table 4 can prevent some important plan updates to be received by the GCS which can result in some accidents or an unsafe behaviour of the assets. The critical requirements generated from the adversary actions are presented in Table 5.

Table 4
Possible adversary actions over manipulation point MP2

Adversary Action	Explanation
H5. Identity spoofing by a malicious IMMS.	GCS receive malicious plans from a malicious IMMS. This action can be done by a nation-state adversary.
H6. Traffic analysis to observe communication patterns.	A hacktivist or curious individuals adversaries can analyse the communication traffic between the IMMS and the GCS to gain a better understanding of how the system works.
H7. Denial of service or jamming attack due to traffic on similar network.	This is an example of an unintentional adversary, where the adversary use similar networks and/or protocols over the same medium, resulting in receiving irrelevant messages or denial-of-service.

7.1. Further Integration of Critical Requirements

In Event-B the critical requirements can be modelled in different forms, such as introducing new invariants, new guards to events as well as introducing new variables and events. Table 6 summarises how the critical requirements identified from the SE-STPA analysis can be formally modelled using Event-B.

From the adversary analysis, two main criteria need to be ensured are confidentiality and integrity. In the subsequent, we present our formal model focusing on these two

Table 5
Critical requirements generation from adversary actions

Hazard	Critical Requirement
H5. Identity spoofing.	CR5. Instructions and control actions must be continuously authenticated, e.g. using unique tokens.
H6. Traffic analysis.	CR6. Communications should be secured against snooping.
H7. Denial of service.	CR7. Only process messages that are in the correct format and are expected.

properties for the communication between the IMMS and GCS (MP2). The model is available at <http://tinyurl.com/JSS2020>.

We start with an abstract machine M0 that models sending and receiving of plans from the IMMS to GCS. At this stage we do not model the network, but focus on ensuring two properties:

- Integrity: The current plan for each GCS must be intended for that GCS.
- Confidentiality: A GCS only knows the plans intended for that GCS.

These properties secures the communications between the IMMS and the GCS and ensure that the generated critical requirements CR5 and CR6 in Table 5 are satisfied. In Event-B, these properties are ensured by the following invariants, where the variables are explained in Figure 7. In Event-B, all events must maintain the invariants.

@inv_integ: $\text{current}^{-1} \subseteq \text{dest}$

@inv_conf: $\text{knows}^{-1} \subseteq \text{dest}$

Here, the notation r^{-1} denotes the inverse of relation r .

We model the network communication in an independent model, M_SymEnc, where we define an encryption function and model sending and receiving of data with and without encryption. Next we refine M0, the previous proved properties will continue to hold by proving refinement.

To illustrate the needs for encryption to ensure integrity and confidentiality, we perform two separate refinements: one refinement is insecure and allows an intruder to know the sent data M1_Plain. M1_Plain includes M_SymEnc but synchronises with the sending and receiving events that do not use encryption/decryption of data (send_plain, receive_plain). Another secure refinement M1_SymEnc uses encryption to prevent an intruder from knowing the data. Machine M1_SymEnc also includes M_SymEnc, but sending and receiving events synchronise with the events that apply encryption and decryption (encrypt_and_send, receive_and_decrypt). The intruder event will synchronise with the same receiving events.

On the one hand, in the insecure case, our formal model results in an unprovable proof obligation. This is because the intruder will be able to know the plan sending to the GCS, hence break confidentiality. On the other hand, in the secure case, the intruder will not know the data without the key, it

can still intercept the data, but cannot decrypt without the key. Figure 7 represents the different Event-B machines and their relationships.

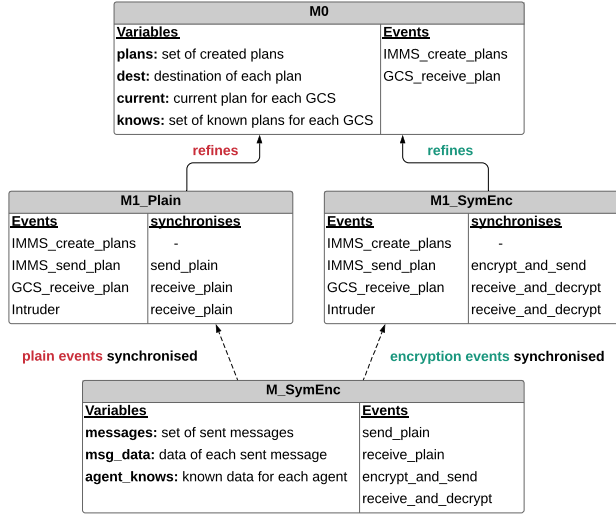


Figure 7: Adversary modelling: Event-B machines relationship

The advantage of using inclusion mechanism is that, we can simply change the encryption function without breaking the refinement dependency. In $M1_SymEnc$, we have used symmetric key encryption M_SymEnc , but we can replace this with other encryption mechanism, e.g., using public keys. Further more, this model of communication can also be reused by other models, for example to model the communication network between GCS and the assets (MP3).

7.2. Causal Factors Analysis

Causal factors include checking the hazards and how they might happen in the control loop. However, causal factors analysis is less concerned with adversarial modelling because the adversary actions are intentional. Therefore, this step can possibly be moved before the adversarial modelling, but it is useful to be done after the integration of the critical requirements in the formal model. In our approach we advocate modelling via decomposition, so may be it is worth splitting the requirements integration into the formal model into two steps, one before the adversarial modelling and one after that.

If we take for example H1 in Table 1, where an asset can deviate from a plan. One of the possible reasons could be a sudden dynamic obstacle that cannot be avoided quickly enough by an update. In this case the asset can deviate from the plan, so we can have the following requirements:

- An asset shall deviate from the plan to avoid a hazardous situation.
- The IMMS shall be informed of any plan deviations.

Another possible cause for deviation from a plan is loss of communication. For example, an update is sent by the

Table 6

Formal modelling of critical requirements

Critical Requirement	Formal Model
CR2. Plans containing no-go zones or hazardous locations shall not be validated and sent to assets.	Only valid plans are approved and sent. This is modelled via invariant and guard in the event <code>oper_approvePlan</code> in <code>oper_imms</code> machine and as a guard in the <code>imms_sendPlan</code> of the combined model.
CR3. A mission can only start if plans are generated and validated.	A mission can only start after a minimum criteria is met, this is discussed in early requirements we strengthen the guards of <code>start_mission</code> event by including valid plans to assigned assets.
CR4. Plan generation can only start after all mission goals are confirmed.	In Event-B we define all mission goals in one event which is disabled by a guard once set. the <code>generate_plan</code> event has a precondition modelled by an event guard to execute only after mission goals are set.
CR5. Instructions and control actions must be accompanied by unique tokens.	Use symmetric key encryption to secure messages using machine inclusion and event synchronisation.
CR6. Communications should be secured against snooping.	Introduce encryption and decryption via synchronised events.
CR7. Only process messages that are in the correct format and are expected.	This cannot be modelled in Event-B, but can be ensured during implementation.

IMMS but never received by the GCS or the asset cannot receive the update sent by the GCS. This could be the result of a communication failure or a denial of service due to an adversary attack. To mitigate for this loss of communication, the asset should follow a predetermined recovery plan. For example the UAV can land in a safe predetermined recovery point, the USV can hold station in its position until communication is re-established, the UUV can abort its mission and resurface.

These requirements and mitigations can be integrated in the system design and can be assessed in the next iteration of the process presented in Figure 3.

8. Conclusions

In this paper, we have proposed an approach for eliciting requirements for autonomous missions and formalising the critical requirements as Event-B models. This is part of the functional process for an Integrated Mission Management

for heterogeneous autonomous systems. Figure 3 illustrates our overall process shows how we augment the requirements through continuous analysis. The proposed approach is iterative where we continuously need to do reviews which can influence the formal modelling on one hand and the formal modelling can influence the system requirements by identifying new requirements, removing ambiguities and defects and ensuring the consistency of the requirements. We also apply an STPA-based approach mainly to generate the critical requirements, but also as a means to analyse the control actions requirements.

The main limitation in our approach is the need to have expertise in Event-B or formal methods in general. However this is inevitable when trying to build a strong safety argument. Another limitation is that STPA analysis is qualitative and lacks the quantitative analysis which requires combining it with other analysis techniques. While our approach relies heavily on formal modelling and STPA, it still has its advantages:

- The combination of formal modelling and STPA helps to discover inconsistencies and gaps in requirements. Using STPA with formal modelling enables the domain experts to assess the system where STPA provides a robust traceability of the critical requirements and their associated formal representation which is crucial for V&V. On the other hand formal modelling gives the critical requirements a precise syntax where the consistency of the constraints can be formally verified.
- The Original SE-STPA approach focused on adding new generated critical requirements as a new Event-B refinement, this is not always possible and can be cumbersome especially when requirements need to be updated as a result of reiteration. In this paper we have focussed on separation of concerns and decomposition and composition of formal models to address issues of re-usability and complexity of formal modelling. For this we have shown how we applied machine inclusion and policing functions and different models that focus on different problems, while we have a high level model of the overall system.
- The separation of concerns also allows us to change the algorithms used without invalidating the formal model. For example, the planning algorithm can be changed, but the same policing function can be applied. We can change the encryption method, but our integrity and confidentiality requirement is still covered and the overall system model still holds and does not need to be reverified. This is in particular very important for safety certification, if this approach was used to build a safety case this does not invalidate our safety case if we update the algorithms.

Traceability between the Event-B formal models and the critical requirements generation using SE-STPA is done manually. Having a tool that supports traceability between the

analysis technique and the formal modelling can be useful to have a fully integrated methodology.

The IMMS software has been written in Python based on our formal models and tested in a trial involving an asset from the three different domains. However, the transformation from Event-B to Python has been done manually¹. In the future, we would like to extend our process of requirements elicitation to include automatic code generation from Event-B as well as test case generation.

In this paper we have also characterised the safety property of the IMMS, which we defined as avoiding no-go zones as a policing function to ensure the safety of the system. In the future we would like to characterise additional safety and security envelopes, in particular looking at more dynamic maritime factors.

A. List of Abbreviations

AS	Autonomous Systems
CPS	Cyber-Physical Systems
GCS	Ground Control Station
IMMS	Integrated Mission Management System
MAS	Maritime Autonomous Systems
Rodin	Rodin platform
SE-STPA	Security-Enhanced Systems-Theoretic Process Analysis
STPA	Systems Theoretic Process Analysis
UAV	Unmanned Aerial Vehicle
USV	Unmanned Surface Vehicle
UUV	Unmanned Underwater Vehicles
UxV	Unmanned Vehicle
V&V	Verification and Validation

References

- Abdulkhaleq, A., Wagner, S., Leveson, N., 2015. A comprehensive safety engineering approach for software-intensive systems based on stpa. *Procedia Engineering* 128, 2 – 11. URL: <http://www.sciencedirect.com/science/article/pii/S1877705815038588>, doi:<https://doi.org/10.1016/j.proeng.2015.11.498>. proceedings of the 3rd European STAMP Workshop 5-6 October 2015, Amsterdam.
- Abrial, J.R., 2010. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press.
- Abrial, J.R., Butler, M., Hallerstede, S., Hoang, T., Mehta, F., Voisin, L., 2010. Rodin: An open toolset for modelling and reasoning in Event-B. *Software Tools for Technology Transfer* 12, 447–466.
- Bensaci, C., Zennir, Y., Pomorski, D., 2018. A Comparative Study of STPA Hierarchical Structures in Risk Analysis: The Case of a Complex Multi-Robot Mobile System, in: 2018 2nd European Conference on Electrical Engineering and Computer Science (EECS), pp. 400–405.
- Bogdiukiewicz, C., Butler, M., Hoang, T.S., Paxton, M., James, H.S., Waldron, X., Wilkinson, T., 2017. Formal development of policing functions for intelligent systems, in: 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE), IEEE. Associated publication: Bogdiukiewicz, C., Butler, M., Hoang, T. S., Paxton, M., Snook, J. H., Waldron, X., & Wilkinson, T. (2017). Formal development of policing functions for intelligent systems. In 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE), IEEE. DOI: 10.1109/ISSRE.2017.40.

¹ We have not shown the transformation from Event-B to Python in this paper because the main focus of this paper is requirements elicitation

- Colley, J., Butler, M., 2013. A formal, systematic approach to stpa using event-b refinement and proof, in: 21th Safety Critical System Symposium (01/02/13). URL: <https://eprints.soton.ac.uk/352155/>.
- Dghaym, D., Turnock, S., Butler, M., Downes, J., Hoang, T.S., Pritchard, B., 2019. Developing a framework for trustworthy autonomous maritime systems, in: International Seminar on Safety and Security of Autonomous Vessels (ISSAV): ISSAV 2019. URL: <https://eprints.soton.ac.uk/434545/>.
- Friedberg, I., McLaughlin, K., Smith, P., Laverty, D., Sezer, S., 2017. Stpa-safesec: Safety and security analysis for cyber-physical systems. *Journal of Information Security and Applications* 34, 183 – 196. URL: <http://www.sciencedirect.com/science/article/pii/S2214212616300850>, doi:<https://doi.org/10.1016/j.jisa.2016.05.008>.
- Hata, A., Araki, K., Kusakabe, S., Omori, Y., Lin, H., 2015. Using hazard analysis stamp/stpa in developing model-oriented formal specification toward reliable cloud service, in: 2015 International Conference on Platform Technology and Service, pp. 23–24.
- Hoang, T., 2013. An introduction to the Event-B modelling method, in: Industrial Deployment of System Engineering Methods. Springer-Verlag, pp. 211–236.
- Hoang, T.S., Dghaym, D., Snook, C., Butler, M., 2017. A composition mechanism for refinement-based methods, in: 2017 22nd International Conference on Engineering of Complex Computer Systems (ICECCS), pp. 100–109. doi:[10.1109/ICECCS.2017.27](https://doi.org/10.1109/ICECCS.2017.27).
- Howard, G., Butler, M., Colley, J., Sassone, V., 2019. A methodology for assuring the safety and security of critical infrastructure based on STPA and Event-B. URL: <https://eprints.soton.ac.uk/429899/>, doi:[doi:10.1504/IJCCBS.2019.098815](https://doi.org/10.1504/IJCCBS.2019.098815).
- Leffingwell, D., 1997. Calculating the Return Investment from more Effective Requirements Management. *American Programmer*, 10, 13–16.
- Leuschel, M., Butler, M., 2008. ProB: An automated analysis toolset for the B method. *Software Tools for Technology Transfer (STTT)* 10, 185–203.
- Leveson, N.G., Thomas, J.P., 2018. STPA Handbook. Cambridge, MA USA.
- MacDonell, S.G., Min, K., Connor, A.M., 2014. Autonomous requirements specification processing using natural language processing. *CoRR abs/1407.6*. URL: <http://arxiv.org/abs/1407.6099>, arXiv:[1407.6099](https://arxiv.org/abs/1407.6099).
- Omitola, T., Rezazadeh, A., Butler, M., 2019. Making (implicit) security requirements explicit for cyber-physical systems: A maritime use case security analysis, in: Anderst-Kotsis, G., Tjoa, A.M., Khalil, I., Elloumi, M., Mashkoo, A., Sameting, J., Larrucea, X., Fensel, A., Martinez-Gil, J., Moser, B., Seifert, C., Stein, B., Granitzer, M. (Eds.), *Database and Expert Systems Applications*, Springer International Publishing, Cham. pp. 75–84.
- Punnoose, R.J., Armstrong, R.C., Wong, M.H., Jackson, M., 2014. Survey of Existing Tools for Formal Verification. Technical Report. Sandia National Lab. (SNL-CA), Livermore, CA (United States). doi:[10.2172/1166644](https://doi.org/10.2172/1166644).
- Thomas, J., Leveson, N., 2013. Generating Formal Model-Based Safety Requirements for Complex, Software-and Human-Intensive Systems, in: *Proceedings of the Twenty-first Safety-Critical Systems Symposium*, Bristol, UK, Safety-Critical Systems Club.
- Young, W., Leveson, N., 2014. Inside risks an integrated approach to safety and security based on systems theory: Applying a more powerful new safety methodology to security risks. *Communications of the ACM* 57, 31–35. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84893411630&doi=10.1145%2F2556938&partnerID=40&md5=07efb2984b5cf13de1fe2cb1583b7d27>, doi:[10.1145/2556938](https://doi.org/10.1145/2556938).
- Zhou, Z., Zi, Y., Chen, J., An, T., 2019. Hazard Analysis for Escalator Emergency Braking System via System Safety Analysis Method Based on STAMP. *Applied Sciences* 9. URL: <https://www.mdpi.com/2076-3417/9/21/4530>, doi:[10.3390/app9214530](https://doi.org/10.3390/app9214530).