Anticipating accidents through reasoned simulation

Citation for published version:

Digital Object Identifier (DOI):
10.1145/3597512

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Publisher's PDF, also known as Version of record

Published In:
TAS '23: Proceedings of the First International Symposium on Trustworthy Autonomous Systems

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Anticipating Accidents through Reasoned Simulation

Craig Innes  
School of Informatics, University of Edinburgh  
Edinburgh, Scotland, UK

Andrew Ireland  
School of Mathematical and Computer Sciences,  
Heriot-Watt University  
Edinburgh, Scotland, UK

Yuhui Lin  
School of Mathematical and Computer Sciences,  
Heriot-Watt University  
Edinburgh, Scotland, UK

Subramanian Ramamoorthy  
School of Informatics, University of Edinburgh  
Edinburgh, Scotland, UK

ABSTRACT
A key goal of the System-Theoretic Process Analysis (STPA) hazard analysis technique is the identification of loss scenarios — causal factors that could potentially lead to an accident. We propose an approach that aims to assist engineers in identifying potential loss scenarios that are associated with flawed assumptions about a system’s intended operational environment. Our approach combines aspects of STPA with formal modelling and simulation. Currently, we are at a proof-of-concept stage and illustrate the approach using a case study based upon a simple car door locking system. In terms of the formal modelling, we use Extended Logic Programming (ELP) and on the simulation side, we use the CARLA simulator for autonomous driving. We make use of the problem frames approach to requirements engineering to bridge between the informal aspects of STPA and our formal modelling.

CCS CONCEPTS
• Theory of computation → Logic and verification; Automated reasoning; Computing methodologies → Model verification and validation.

KEYWORDS
Autonomous systems, Hazard analysis, Formal modelling, Simulation.

ACM Reference Format:

1 INTRODUCTION
System-Theoretic Process Analysis (STPA) [24] extends traditional hazard analysis techniques to include a range of causal factors that can affect system safety. A key goal of STPA is the identification of causal factors that give rise to unsafe control actions, which in turn could lead to hazards and ultimately an accident — what Nancy Leveson calls a loss scenario.

While STPA is systematic, it represents a relatively informal style of analysis. As a consequence, STPA provides no mechanized support to prevent errors such as inconsistencies and invalid assumptions entering into the analysis. Given that loss scenarios often arise because of invalid environment assumptions [24], we believe that mechanized support for STPA is well motivated.1

We propose a mechanized approach that uses formal modelling to verify the consistency between safety related constraints and assumptions. Such a verification relies upon the validity of the safety assumptions. To check validity we use simulation. Specifically, we search for environment conditions under which the safety assumptions are violated. Such violations represent loss scenarios. So while formal modelling ensures consistency of the analysis, we rely upon simulation to identify the concrete loss scenarios.

Our proposed approach is at the proof-of-concept stage. We illustrate it using a case study based upon a simple car door locking system. To bridge between STPA and formal modelling we use the notion of a problem frame [20] combined with Extended Logic Programming (ELP) [13], while on the simulation side, we use the CARLA (Car Learning to Act) simulator [10] for autonomous driving. In section 2, we provide background on STPA and the aspects of ELP and problem frames that we make use of in this paper. We also provide an overview of the CARLA simulator and the features that we rely upon in this paper.

2 BACKGROUND
We follow the 4-step description of STPA given in [21]. The first step involves identifying the potential losses (L), system-level hazards (H) and system-level constraints (SC). A hazard denotes a system state coupled with environmental conditions that lead to a potential loss. The role of the system-level constraints is to guard against the identified hazards from occurring. Step 2 focuses on the achievement of the system-level constraints. First, it involves defining a control structure – a diagrammatic description of a system’s controllers and controlled processes, together with how they interact, i.e., control actions and feedback. Second, the responsibilities for each controller are defined, where a responsibility (R) represents a refinement of a system-level constraint. Each responsibility is

1Such flaws can occur at design-time but can also arise as a consequence of changes to a system’s operational environment.
identified with part of a controller’s process model and control logic, which are informed by feedback from the processes that it controls. Finally, a control action is associated with each responsibility. Step 3 identifies how inadequate control could lead to a hazard by considering the following cases:

1. A control action that when not applied causes a hazard.
2. A control action that when applied causes a hazard.
3. A control action that when applied too early, too late or out of sequence causes a hazard.
4. A control action that is stopped too soon or applied too long causes a hazard.

Note that the final case is applicable to non-discrete control actions. For each unsafe control action, a corresponding control constraint is defined, which specifies a property that must be satisfied in order to prevent the controller exhibiting the unsafe control actions. Figure 1 summarises the first 3-steps of STPA. Despite design-level verification, the fourth step focuses on identifying how unsafe control actions could occur and lead to the violation of system-level constraints. As noted above, these are referred to as loss scenarios. In [24], the process of identifying a loss scenario is described as ‘the usual “magic” one that creates the contents of a fault tree.’ There are many ways in which unsafe control actions could occur. In section 4.4, we focus on flawed assumptions as the source of loss scenarios.

As mentioned above, we use the notion of a problem frame [20] to bridge between the informal and the formal. A problem frame provides a way of verifying the consistency between a system-wide requirement (i.e., system-level constraint) and the specification of a software component (i.e., control constraint). The problem frame representation has both graphical and logical elements. Substituting the STPA control structure with a problem frame involves formalizing the domain assumptions associated with the system and its operational environment. An informal verification is represented by means of an informal argument diagram [30] which will be illustrated in sections 4.3 and 4.4.

In terms of formal logic, the problem frames approach is relatively agnostic. Our case study is quite simple, so we opt for classical propositional logic – specifically we use Extended Logic Programming (ELP) [13]. As a consequence, we ignore delays between observations and actions. More expressive formalisms could be used. ELP represents an extension of logic programming, e.g., Prolog [7, 15]. Logic programming is a declarative programming paradigm that is based upon formal logic. A program is defined in terms of a collection of clauses, where a clause is either a fact or a rule. This clausal form, which is a subset of first-order logic, takes the following general form:

\[ L: \neg A_1, \ldots, A_m, \neg B_1, \ldots, \neg B_n \]  

where \( A_i \) is a positive predicate and \( B_j \) is a negated one. Program execution is based upon the resolution inference mechanism. From a reasoning perspective, logic programming is non-monotonic. That is, the fact that \( \Gamma \vdash C \) holds does not guarantee that \( \Gamma, A \vdash C \) also holds. This is because negation (i.e., \( \neg \)) is evaluated by a non-monotonic rule called negation-as-failure (a form of the closed-world assumption). This means that if a statement cannot be derived then it is assumed to be false. ELP extends logic programming with a classical (or explicit) form of negation. While \( \neg P \) denotes negation-as-failure, \( \neg P \) denotes classical negation. To conclude \( \neg P \) it is not sufficient to have failed to derive \( P \); explicit evidence for the negation of \( P \) is required.

Formal modelling is suited to relatively abstract models where the state space is small and can be exhaustively explored. In contrast, simulation allows one to selectively explore large state spaces. In terms of simulations, we have used CARLA [10], which is a simulation platform for testing the performance of automated vehicle systems. It provides a rich set of features for capturing the many complex interactions between the many integrated modules within a modern Autonomous Vehicle (AV) pipeline. In terms of physical dynamics and control, CARLA can simulate vehicles that are affected by a highly granular set of parameters. This can be as broad as overall mass, all the way down to gear ratio, drag coefficient of the chassis, tyre friction, and wheel stiffness. Also, CARLA provides libraries, e.g., Python API, to specify/program scenarios for both local control and global route planning of large numbers of vehicles on the road simultaneously. A simulator provides a way to efficiently generate many thousands of example rollouts/scenarios of the AV system given a user-specified starting specification. So, while symbolic domain knowledge may provide heuristics about which configuration parameters are likely to be relevant to safety, testing variants of those parameters out in simulation allows us to see how they affect behaviour in practice, and thus search for falsifying examples.

3 RELATED WORK

As illustrated above, the problem frame approach provides a technique for verifying the correctness of a system-wide requirement with respect to a specification and domain assumptions. In [30] Alloy [18] is used to formalize and verify problem frame instances. Alloy uses first-order relational logic. Moreover, they use a technique called requirements progression to mechanically derive a specification from a given system-wide requirement and its associated domain assumptions. Here this would correspond to deriving control constraints from the system-level constraints and associated domain assumptions. In section 4.4, we use the notion of an anti-system-level constraint. This is similar to the notion of an anti-requirement introduced in [25] to represent the intentions of a malicious user.

As a logic-based declarative language, ELP has been used as a formal modelling and problem-solving tool. ELP is supported by a knowledge representation and reasoning framework called Answer Set Programming (ASP) [12, 26]. ASP has been applied to decision-making, planning, diagnostic reasoning and configuration optimisation, e.g., decision support systems [27], configuration optimisation in railway safety systems [4], action planning in the setting of multiple robot collaboration tasks [11], and diagnosing failures of an automatic whitelisting system [6].

Combining formal reasoning and simulation is explored in [14], where the SAL model checker is used to guide the use of agent-based simulation via WMC [31]. Specifically, the SAL model checker was used to model a known automation surprise associated with the Airbus A320 autopilot, which was then subsequently explored via WMC. They found that WMC provided psychological plausibility to the counterexample generated by SAL. Moreover, WMC provided
Anticipating Accidents through Reasoned Simulation

The diagram above shows the first 3 steps of the STPA hazard analysis process – from the definition of losses and hazards through to defining control actions and identifying how unsafe control actions can lead to hazards. Step 4, which is not shown in the diagram, focuses on the identification of loss scenarios (see section 4.4).

Figure 1: Summary of steps 1-3 of STPA.

Simulating autonomous vehicle behaviour requires coordinating multiple related, high-fidelity (computationally expensive) components. This includes sensor models (e.g., Cameras, Lidar, Radar), multibody vehicle dynamics, traffic flows, and pedestrian behavioural patterns [33]. In particular, such platforms distinguish between simulating the environment itself, and the environment as seen by the vehicle’s perception systems. Discrepancies between these views can be dangerous [16]. Such simulators constitute a core component in many verification tasks such as falsification and failure probability estimation [9]. AV Simulators allow us to efficiently replicate a given safety scenario with high fidelity, but it is impossible to test all possible road scenarios. Thus, the primary challenge is to identify which scenarios are worth simulating. Existing approaches to scenario selection involve either hand-crafted databases of safety-critical situations or data-driven scenario generators trained on large amounts of passively collected real-world traffic data [32].

4 ADDING FORMAL MODELLING AND SIMULATION TO STPA

To illustrate our proposal, we use an automatic car door locking example that involves a simple interlock mechanism. The example is inspired by Michael Jackson’s aircraft braking system example [19], and is strongly related to an accident involving a fly-by-wire aircraft [22]. In our example, we are concerned with ensuring that while a car is moving its doors should be automatically locked and while the car is stationary, the doors should be automatically unlocked. The task is to construct a Door Lock Control Unit (DLCU) that ensures the locking and unlocking of the doors occurs at the right time. A fundamental design assumption will be that when the car is moving the wheels will be turning. From the perspective of STPA, the controlled process involves the locking mechanism for the car’s doors and the car’s wheel sensors. The DLCU receives feedback in the form of wheel-pulse signals from the sensors when the wheels are turning. In terms of control, the DLCU will be able to send lockDoors and unlockDoors signals to the doors. Below we follow the first three steps of the STPA hazard analysis technique as described above. We use the following propositions to bridge between the informal and formal:

WT: true when car wheels are turning, otherwise false.
DL: true when car doors are locked, otherwise false.
WP: true when there exists a wheel-pulse signal, otherwise false.
DS: true when there exists a door-lock signal, otherwise false.
MV: true when the car is moving, otherwise false.
We have associated the following two hazards to...of their lives or injuries due to safety failings would be personally catastrophic/devastating. In addition, the manufacturer would experience loss through the legal consequences of such safety failings, as well as potentially suffering reputational loss. For the purposes of the example, we will focus solely on the human perspective, i.e.,

L-1: loss of life or injury.

We have associated the following two hazards to L-1:

H-1: a passenger opens their door while the car is moving.
H-2: a passenger is unable to open their door while the car is not moving.

H-1 may result in a person falling from a moving car leading to loss of life or injury, while H-2 may prevent a person from leaving a stationary car when their safety is in jeopardy, e.g., the car is on fire. Following STPA, each hazard should be associated with a system-level constraint, i.e.,

SC-1: if the car is moving then the doors must be locked (MV ⇒ DL). [H-1]
SC-2: if the car is not moving then the doors must be unlocked (∼MV ⇒ ∼DL). [H-2]

Note that composing the above two system-level constraints gives:

SC-3: the car is moving if and only if the doors are locked (MV ⇔ DL). [H-1, H-2]

Note that we will use SC-3 for the purposes of verification and the identification of loss scenarios.

4.2 Model the control structure (step 2)

As noted in section 2, we use the notation of a problem frame rather than a control structure in order to bridge between the informal and the formal. Figure 2 provides a problem frame for DLCU. Note that the problem frame makes explicit the domain assumptions and shared phenomena that logically link the system-level constraint with the control constraint. Note in particular that the design assumption that the movement of the car is equivalent to its wheels turning is explicitly represented.

In terms of responsibilities, DLCU has two; each represents a refinement of a system-level constraint, i.e.,

R-1: enable door locks when the car is moving. [SC-1]
R-2: disable door locks when the car is not moving. [SC-2]

As described in section 2, each responsibility is associated with elements of a controller’s process model and control logic. These associations for DLCU are given below. They include both the STPA informal descriptions as well as a logical counterpart:
The above problem frame contains 3 domains, i.e., DLCU, Car and Road, where the Car domain has 2 subdomains, Wheels and Doors. The Road domain captures a key design assumption, i.e., \((WT \Leftrightarrow MV)\). Domains interact via shared phenomena, e.g., DLCU and Wheels share wheel-pulse signals (WP). The DLCU is a special domain; it represents the software controller, where its intended behaviour is specified via the control constraint CC-3. The dashed ellipse denotes the required system-level constraint, i.e., SC-3. The dashed lines indicate that the system-level constraint relates to the Road and Doors domains.

**Figure 2: A Problem Frame for DLCU.**

<table>
<thead>
<tr>
<th>Control action</th>
<th>Not applied causes a hazard</th>
<th>Applied causes a hazard</th>
<th>Applied too early, too late or out of sequence causes a hazard</th>
<th>Stopped too soon or applied too long causes a hazard</th>
</tr>
</thead>
<tbody>
<tr>
<td>lockDoors (CA-1)</td>
<td>Doors not locked when car moving (H1)</td>
<td>Doors locked when car stationary (H2)</td>
<td>Doors locked too late (H1)</td>
<td>N/A</td>
</tr>
<tr>
<td>unlockDoors (CA-2)</td>
<td>Doors locked when car stationary (H2)</td>
<td>Doors unlocked when car moving (H1)</td>
<td>Doors unlocked too soon (H1)</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 1: Identifying Hazardous System Behaviour for DLCU.

As shown in Figure 4, replacing assumption (4) by (5) (within the Road domain) breaks the verification proof given in Figure 3. In Figure 5, we present an alternative formal model of the loss scenario. We replace SC-3 by an anti-system-level constraint, i.e., anti-SC-3. While a system-level constraint specifies, ‘something bad will never happen’, an anti-system-level constraint specifies, ‘something bad will happen’. The idea is that a scenario that achieves an anti-system-level constraint will denote a loss scenario.

So formal modelling provides a mechanism for generating abstract loss scenarios. However, it provides no insight into the feasibility of the generated loss scenarios. We use simulation to provide such insight. Simulators are rich in terms of their knowledge about real-world phenomenon. What we propose is to leverage this knowledge in order to elaborate an abstract loss scenario. In so doing, we are exploiting synergies that exist between formal modelling and simulation, and providing engineers with more realistic and accessible loss scenarios to consider.

In our car example, (6) provides the starting point for a simulation. Bridging the gap between the formal modelling (i.e., ELP) and simulation (i.e., CARLA) requires mapping the propositions \(MV\) and \(WT\) onto the corresponding parameters within the CARLA simulation space. Simply equating \(MV\) with the car’s velocity and \(WT\) with wheel speed is not sufficient. That is, it gives a low-fidelity approximation of the car’s behaviour. CARLA provides various
Clauses C1 and C2 define how SC-3 can be violated. Note that the satisfaction of –dl and –mv requires explicit evidence for the negated propositions, i.e. failure to satisfy dl and mv is not sufficient to conclude that their negations hold. Clause C3 defines an explicit violation in terms of the failure satisfy violation. Clause C4 encodes the top-level proof: explicit evidence that there are no violation implies SC-3 it true. The Extended Logic Programming demos are available at [28].

Figure 3: Proving consistency between CC-3 and SC-3.

mechanisms to assist a developer to define a simulation. In terms of the simulation parameters, these are classified as follows:

- **Physics of a vehicle**: includes concepts such as torque, drag and friction.
- **Traffic behaviour**: predefined behaviours are provided, i.e., cautious, normal, aggressive, but a developer is free to define the behaviour of the vehicles within their simulation.
- **Perception**: weather conditions and the attributes of sensors, e.g., accuracy in poor lighting.

At the level of the physical parameters, the following are relevant to our example:

- Frictional coefficient of the road surface
- Velocity/acceleration of the vehicle in previous time step
- Tire friction
- Wheel torques
- Wheel speeds
- Engine RPM
- Drag coefficient on the vehicle body
As shown in Figure 4, replacing assumption (3) by (4) (within the Road domain) breaks the verification proof given in Figure 3. In Figure 5, we present an alternative formal model of the loss scenario. We replace SC-3 by an anti-system-level constraint, i.e., anti-SC-3. While a system-level constraint specifies, 'something bad will never happen', an anti-system-level constraint specifies, 'something bad will happen'. The idea is that a scenario that achieves an anti-system-level constraint will denote a loss scenario.

The consequences of violating the design assumption, i.e., \((WT \iff MV)\), are shown above. Firstly, the informal argument diagram shows where the verification proof breaks. Secondly, the violation is represented as a sequent proof. Lastly, the ELP formulation of the violation is given.

The case study explored the 8-dimensional factors listed above. Here, we combined the guidance provided by the formal model, i.e., \(\neg WT \land MV\), and the relevant physical parameters from CARLA, to construct a potential loss scenario that violates (4). The violation occurs when the velocity is greater than 0 and the frictional coefficient of the road surface changes from 1.0 to be less than or equal to 0.5, i.e. the car is skidding.

Here the mapping was hand-crafted. To fully realize our proposal additional guidance is required in order to constrain CARLA. That is, we need access to domain knowledge about how stuff fits together as well as how stuff behaves in accordance with the laws of physics. For our illustrative example the concept of friction plays a key role in the loss scenario, i.e. wheel rotation requires friction between the car’s tyres and the road surface. Such knowledge could be represented in terms of an ontology.

From the perspective of the case study, our use of CARLA is summarised in Figure 6. Note that the simulation demo and setup is available via [29], which includes animations generated by CARLA. In Figure 7, a snapshot of the CARLA simulation development environment is provided.

Figure 4: A violation of the consistency between CC-3 and SC-3.
Above the anti-system-level constraint, i.e., $\neg DL \Rightarrow \neg MV$, is shown to be provable if the original design assumption, i.e., $WT \Rightarrow MV$, is replaced by $MV \Rightarrow \neg WT$. The first proof takes the form of an informal argument diagram. In addition, a sequent proof is provided along with the corresponding ELP formulation.

\[
\neg WP \Rightarrow \neg DS \quad \neg DS \Rightarrow \neg DL \\
MV \Rightarrow \neg WT \quad \neg WT \Rightarrow \neg DL \\
MV \Rightarrow \neg DL
\]

Figure 5: Proving consistency between CC-3 and anti-SC-3.

5 Future Work

Two significant gaps need to be addressed in our future work. In order for our proposed approach to be effective and accessible, assistance is needed in bridging between STPA and ELP as well as ELP and CARLA. Tool support will be essential in managing the details. Full automation is not feasible or desirable. We envisage a level of dialogue between the tool and the user in order to ensure that each step in the translation is valid. In general, we believe that Interactive Task Learning [3, 23] provides a promising approach to bridging these gaps. We are exploring mechanical analysis and vehicle models to develop hierarchical abstractions in order to bridge the gap between formal modelling statements and simulation variables, e.g., car moving ($MV \Rightarrow \neg DL$), is replaced by ($MV \Rightarrow \neg WT$). The first proof takes the form of an informal argument diagram. In addition, a sequent proof is provided along with the corresponding ELP formulation.

\[
\neg WP \Rightarrow \neg DS \quad \neg DS \Rightarrow \neg DL \\
MV \Rightarrow \neg WT \quad \neg WT \Rightarrow \neg DL \\
MV \Rightarrow \neg DL
\]

Above the anti-system-level constraint, i.e., $(MV \Rightarrow \neg DL)$, is shown to be provable if the original design assumption, i.e., $(WT \Rightarrow MV)$, is replaced by $(MV \Rightarrow \neg WT)$. The first proof takes the form of an informal argument diagram. In addition, a sequent proof is provided along with the corresponding ELP formulation.

\[
\neg WP \Rightarrow \neg DS \quad \neg DS \Rightarrow \neg DL \\
MV \Rightarrow \neg WT \quad \neg WT \Rightarrow \neg DL \\
MV \Rightarrow \neg DL
\]

**Figure 5: Proving consistency between CC-3 and anti-SC-3.**
The inputs above on the left include, i) setup parameters for the simulation, and ii) template scenarios that encode the abstract loss scenario identified by the formal modelling. Note that while a graphical notation is shown, the developer uses a Python API to specify the inputs. The output from the simulator takes the form of i) an animation of the loss scenario, and ii) a log of all the relevant parameters. Note that a demo of the case study simulation is available via [29], which includes the animations generated by CARLA.

Figure 6: Exploring loss scenarios using simulation.

6 RESPONSIBLE RESEARCH AND INNOVATION

Where innovation has safety related concerns, gaining public acceptance is an important challenge that needs to be addressed. This is particularly true in the case of autonomous systems, such as self-driving cars. We believe that building public confidence in the way in which a system is designed will make a valuable contribution to addressing this challenge. Safety related defects are often introduced because of gaps between the traditional stages of the development process [5]. Our work seeks to reduce these gaps and therefore reduce the risk of such defects occurring. This argument requires a dialogue with the public, where evidence in the form of case studies will play a crucial role.

7 CONCLUSION

STPA is a popular technique that aims to assist engineers in analysing a system’s control actions in relation to system-level hazards. The technique is structured and systematic but relatively informal compared to a formal method. Problem frames, if formalized, provides a mechanism for strengthening certain key informal aspects of STPA, i.e., verifying the consistency between safety related constraints and assumptions. Moreover, given such a formal verification we suggest that it can be used to identify loss scenarios related to invalid assumptions. Specifically, we propose the use of simulation to search for environment conditions under which safety related assumptions are violated. Engineers must be responsible for any safety related decisions, so ultimately it must be an engineer that judges whether or not a given loss scenario is feasible. What we are proposing is an approach that assists engineers by providing them with simulated scenarios to review that are realistic and accessible.

ACKNOWLEDGMENTS

The research reported in this paper is supported by the UKRI Trustworthy Autonomous Systems Node in Governance and Regulation (EPSRC grant EP/V026607/1).

REFERENCES
