STPA for Learning-Enabled Systems: A Survey and A New Practice

Yi Qi[1], Yi Dong[1], Siddartha Khastgir[2], Paul Jennings[2], Xingyu Zhao[1,2] and Xiaowei Huang[1]

***Abstract*— Systems Theoretic Process Analysis (STPA) is a systematic approach for hazard analysis that has been used across many industrial sectors including transportation, energy, and defense. The unstoppable trend of using Machine Learning (ML) in safety-critical systems has led to the pressing need of extending STPA to Learning-Enabled Systems (LESs). Although works have been carried out on various example LESs, without a systematic review, it is unclear how effective and generalisable the extended STPA methods are, and whether further improvements can be made. To this end, we present a systematic survey of 31 papers, summarising them from five perspectives (attributes of concern, objects under study, modifications, derivatives and processes being modelled). Furthermore, we identify room for improvement and accordingly introduce DeepSTPA, which enhances STPA from two aspects that are missing from the state-of-the-practice: (i) Control loop structures are explicitly extended to identify hazards from the data-driven development process spanning the ML lifecycle; (ii) Fine-grained functionalities are modelled at the layer-wise levels of ML models to detect root causes. We demonstrate and compare DeepSTPA and STPA through a case study on an autonomous emergency braking system.**

I. INTRODUCTION

Systems Theoretic Process Analysis (STPA) is a hazard analysis method based on System-Theoretic Accident Model and Processes (STAMP) [37], which has been used in aerospace [41], aviation [51], national defence [57], nuclear power [47], as well as other industries [55]. In STAMP, system safety is regarded as a control problem. Uncontrolled external disturbances, component failures and/or abnormal component interactions can result in system accidents. System safety is ensured when the control process adheres to safety constraints [37]. Based on STAMP, STPA begins with a control structure model to evaluate each stage of the function within the control loop and then identify the hazards that can impact the system's dynamic behaviour.

Recently, the use of Machine Learning (ML) to analyse complex data and integrate them into Cyber-Physical Systems (CPSs), known as Learning-Enabled Systems (LESs), has become widespread. When applying LESs in safety-critical domains, safety assurance is essential to their successful deployment and regulatory compliance, which remains a pressing challenge [13], [34], [60] despite recent efforts [7], [8], [29], [20]. Hazard identification is the first and critical step in safety assurance which enables the detection of causes and measures for mitigations of safety risks. Given the popularity of STPA for traditional safety-critical systems, there is a growing interest in utilising STPA for LESs. However, there is no dedicated literature review on this emerging research direction, and the following questions are yet to be answered:

Whether the original STPA method requires adaptation to address new characteristics of ML components? How effective is STPA for LESs? Is there room for improvement?

To answer, we first conduct a systematic survey on the use of STPA for LESs to showcase its recent advancements from five perspectives, i.e., attributes of concern, objects under study, modifications to STPA, derivatives of the analysis, and processes being modelled as control loops. While STPA appears to be a promising technique towards safer LESs, the survey suggests two major gaps in the state-of-the-art:

Gap 1: Most studies only consider the system operation process *after* the deployment. For traditional systems without ML components, operational failure information is normally sufficient to identify root causes and mitigations. However, for LESs, the root causes of ML failures can be in the model architecture, training process, or data collection and cannot be fully understood and detected by only observing operational failures. This view of considering all stages in the ML lifecycle has been presented in the academic discussions on the certification process [30], [8] and the policy level discussions on the governance structure [22], [18].

Gap 2: In all surveyed studies, STPA works at the software/hardware component level, i.e., the minimal granularity of the functionalities modelled by STPA control structures is an entire component. This is sufficient for traditional hardware/software components whose specifications and implementations are well understood. However, for LESs, locating causes at layer-wise level *inside* ML models (e.g., those arising from inputs to a fully-connected layer and feature maps extracted by a convolutional layer) can be more effective in both identifying and mitigating hazards. This aligns with research efforts to unpack complex ML models through visualisation, explanations, verification and testing.

To bridge these two gaps, we introduce DeepSTPA, a novel extension of STPA for LESs. DeepSTPA improves upon STPA in two ways: (i) it explicitly depicts how each stage of the ML lifecycle handles data through control loop structures, to uncover hazards originating from the data-driven development process; (ii) it represents finer functionalities down to the layers of ML models to uncover "deeper" root causes for more effective mitigations of hazards. We demonstrate DeepSTPA through a comparative case study on Autonomous Emergency Braking (AEB) systems.

Contributions of this paper include: *i)* A systematic literature survey devoted to STPA and LESs is conducted, which concludes with positive findings and areas for improvement

[1] University of Liverpool, L69 3BX, U.K. {yi.qi,yi.dong, xingyu.zhao,xiaowei.huang}@liverpool.ac.uk

[2] University of Warwick, CV4 7AL, U.K. {s.khastgir.1, paul.jennings}@warwick.ac.uk

from diverse perspectives. *ii)* Based on identified problems with the state-of-the-art, a new and adapted way of practicing STPA called DeepSTPA is proposed to explicitly consider ML characteristics. *iii)* We demonstrate DeepSTPA with real-world case studies, including a Deep Reinforcement Learning (DRL) based AEB system.

II. Preliminaries

A. Learning-Enabled Systems

LESs are systems that utilise ML algorithms to continuously improve their performance over time [7]. These ML models are data-driven and make predictions/decisions, normally being integrated into CPSs [52]. However, despite their numerous benefits in adapting nonlinearities for complex and high-dimensional tasks, LESs are not immune to failures.

A comprehensive ML lifecycle model can be found in [8], which identifies assurance desiderata for each stage and reviews existing methods that contribute to achieving the desiderata. An ML lifecycle generally consists of four stages—data preparation, model training, evaluation and deployment, as shown in Fig. 1. Instead of a waterfall process, it is normally a spiral process [8] that new data gathered during the operation of existing models can be exploited and, where appropriate, new models may be learnt and deployed.

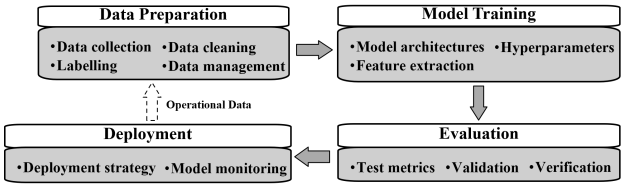

Fig. 1. The ML lifecycle comprises four stages—a spiral process model

Fig. 1 illustrates the process of supervised learning, where data preparation involves multiple steps to preprocess the sensor data. For unsupervised learning and reinforcement learning, some of these steps need adaptation. For example, reinforcement learning requires the environment preparation instead of data preparation, which involves creating the simulation scenarios and defining the reward function.

As an example, DRL is a subfield of ML that combines deep learning and reinforcement learning [6], which is adopted in our later case study. Fig. 2 displays the architecture of DRL which consists of the environment and agent. In DRL, the agent's objective is to learn a policy that maximises the total reward it accumulates over time. The environment interacts with the agent by providing states, receiving actions from the agent, and offering rewards as feedback. The agent's decisions are guided by the observed states and its policy, and the actions it chooses impact the subsequent states and rewards. Through a combination of exploration and exploitation, the agent learns to make informed decisions to optimise its actions for higher rewards within the environment [6]. Deep Q Network (DQN) is a specific DRL algorithm that uses Neuron Networks (NNs) to approximate Q-values, representing expected future rewards for different actions in a given state. [40]. NNs usually consist of several layers, each with specific functionalities, e.g., **convolution layers** implement features extraction, **pooling layers** preserve the main features while reducing parameters and computation, and **fully-connect layers** in which each neuron applies a linear transformation to the input vector through a weights matrix.

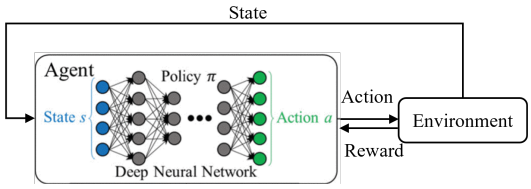

Fig. 2. Schematic structure of deep reinforcement learning

B. Systems Theoretic Process Analysis

In STAMP, three fundamental concepts from systems theory are used—constraints, hierarchical levels of control, and process models. These concepts provide the foundation for analysing complex systems and understanding the interactions and relationships between various components in a system that can lead to accidents [36]. STPA utilises STAMP's control structures [36] to find potential risks or hazards of the whole system, and identify causal scenarios and present safety requirements before the system is put into use. Typically, the STPA workflow includes the following steps [37]: 1) Define Accidents/Hazards. 2) Model Control Structure. 3) Identify Unsafe Control. 4) Identify Causal Scenarios. 5) Derive Safety Requirements.

The first step is to define some accidents or hazards from a high level. Typically, these are serious consequences for personnel or equipment that can be anticipated. The following step is to establish a system control structure that separates the control loop from its components, including the starting point, intermediate processes (e.g., mechanical and software structures), and completion components. Then, potential Unsafe Control Actions (UCAs) can be identified using standard UCAs form with four categories: *(T1) Not providing the control action leads to a hazard*; *(T2) Providing the control action leads to a hazard*; *(T3) Providing a potentially safe control action but too early, too late, or in the wrong order*; and *(T4) The control action lasts too long or is stopped too soon*. The STPA then identifies potential risks/hazards by finding UCAs and determining the possible causes and scenarios. Finally, the STPA provides safety requirements for each potential risk/hazard.

III. STPA for LESs: A Survey

While there is a growing interest in applying STPA for LESs, the absence of a systematic review of emerging papers on this topic motivates the survey.

A. Scope and Paper Collection

In the literature search, we adopted Atlas.ti 6.0 [23], a qualitative research analysis tool to collect papers. The

search function of this survey is presented below, and these keywords only applied to the paper title and abstract.

$$\begin{aligned} Search := &[STPA|STAMP] + [(Learning\ Enabled\ Systems)| \\ &(Robotics\ \&\ Autonomous\ Systems)|(Machine\ Learning)] \end{aligned} \quad (1)$$

where + indicates "AND", and | indicates "OR". Moreover, papers, books and thesis were excluded based on the following criteria: i) not published in English; ii) strictly less than four pages; iii) exists duplicated versions; iv) can not be retrieved using IEEE Explore, Google Scholar, Electronic Journal Center, or ACM Digital Library.

We selected 144 papers using search function (1), then filtered out those that only mentioned STPA in the introduction, related work, and future work sections, leaving 59 papers for further analysis. After careful examination, we further narrowed down our selection to 31 papers by removing duplicates and excluding those that did not combine the two themes of STPA and "learning" as a single topic despite mentioning both sets of keywords in the main methodology and case studies sections, cf. Table I. It summarises the works from five perspectives—*Attributes of Concern*, *Objects Under Study*, *Modification of STPA*, *Derivatives of the Analysis*, and *Processes being Modeled as Control Loops*.

B. Survey Results

a) Attributes of Concern: Safety is the attribute that STPA has traditionally focused on, with the main goal of identifying hazards and enhancing the safety of the applied system. With the advancement of ML, in addition to safety, security and privacy have emerged as the two attributes of growing interest, which encompass guarding against external system breaches and noticing the privacy implications of system usage. This is exemplified by the development of STPA-sec [44], [31], [50], [54], [26] and STPA-priv [39], [53], indicating the flexibility of STPA in terms of accommodating attributes beyond safety. STPA-sec focuses on analysing the vulnerability of the system to external attacks, while STPA-priv highlights the data privacy issues in the system. Reliability is also incorporated in [5] which uses STPA to analyse human activities that affect the risk of human-machine interaction in probabilistic risk assessment.

b) Object Under Study: Autonomous (ground) vehicles are the most commonly studied object by STPA due to their popularity, complexity and pressing need for safety [31], [59], [1], [54], [50], [38], [32], [4], [56], [19], [25], [26], [46]. Autonomous ships and drones are also studied in [58], [49], [27], [61], [5] and [16], respectively. Collaborative robots [11], [3], [14] and Autonomous Mobile Multi-robots [9], [10], [14], [12] are also active areas of applying STPA. Moreover, STPA-priv was initially implemented for smart televisions [53] before being extended to E-health [39], recognising that privacy is a vital concern for humans. Moreover, STPA-sec has also found use in the Aeronautic industry [44]. STPA appears to be effective in analysing such complex CPSs, thanks to its holistic and systematic approach, emphasising causality for early hazard identification.

c) Modification of the Method: While STPA can be directly applied on LESs, due to the intricacy of LESs, many studies advocate combining STPA with other safety analysis methods, e.g., the aforementioned STPA-sec and STPA-priv, the combined use of STPA and Causal Analysis based on System Theory (CAST) [2], a hybrid method for assessing system reliability combines STPA and Success Likelihood Index Method (SLIM) [5], and the combination of STPA with Uncontrolled Flows of Information and Energy (UFoI-E) [27], Bow Tie methodology (Bowtie) [11], Fault tree analysis (FTA) [9] and multilevel run-time monitors [24]. It is also feasible to integrate STPA with hierarchical modelling approaches for intricate systems [50], [31]. STPA may jointly analyse with Analytic Hierarchy Process (AHP) [10] and Stochastic Petri Nets (SPN) [12] respectively and recommend new safety requirements. The advancements have been made in the integration of STPA with domain-specific safety frameworks for autonomous vehicles and ships, e.g. [4], [19], [61]. In [17], STPA is combined with model checking to offer a formal and unambiguous representation of the analysed system. How STPA can make use of Large Language Models like ChatGPT was firstly investigated in [46].

d) Derivatives of the Analysis: Typically, the final step of STPA is to propose safety requirements as a solution to hazards identified. In our survey, it has come to our attention that more diverse use of STPA has been documented. Specifically, corresponding test cases can be created and different results can be obtained by testing the LESs in varying environments [54], [32], [49]. Safety issues can be pinpointed at the system architecture design level, thus mitigation strategies based on new architecture can be derived from STPA [4], [1].

e) Processes Modelled as Control Loops: In most studies, STPA is used to analyse system dynamics during the operation, where interactions between end-users/operators, hardware/software components are modelled as actions and feedback in control loops. However, unlike traditional systems for which operational information may be sufficient for the identification of hazards, causes and mitigations, root causes of ML failures can be located in the data management, ML architecture, training process, etc. Thus, the safety analysis of LESs should also consider activities involving the data owners, providers (who train the ML model), deployers (who deploy the ML model into a larger LES), and the end-users, which requires the modelling of each stage in the ML lifecycle. Moreover, as the ML model becomes more versatile and "deeper", fine-grained functionalities at layer-wise level are introduced. To locate the causes from such lower levels and provide effective mitigations for identified hazards, it may be necessary to model how these layers inside the ML model process input data during the operation. To bridge the two gaps we adapted a new way of practicing STPA, DeepSTPA, in the next section.

IV. An Adapted Practice: DeepSTPA

We develop an adapted practice called DeepSTPA, to cater for LESs with respect to those two gaps we identified in the

TABLE I

SURVEYED PAPERS (**NB**, AVS, ASS, COBOTS, AMRS, ACS, AUVS, RFSS, MRM, OP AND DP REPRESENT AUTONOMOUS VEHICLES, AUTONOMOUS SHIPS, COLLABORATIVE ROBOTS, AUTONOMOUS MOBILE MULTI-ROBOTS, AUTOMATIC CRANES, AUTONOMOUS UNDERWATER VEHICLES, ROBOTIC FLIGHT SIMULATOR, MULTILEVEL RUNTIME MONITORING, OPERATION PROCESS AND DEVELOPMENT PROCESS, RESPECTIVELY)

Year	List	Attributes of concern	Object under study	Modification of the method	Derivative of the analysis	Process Modeled
2015	[2]	Safety	N/A	Yes (XSTAMPP)	Requirements	OP
2016	[53]	Safety, Privacy	Smart TV	Yes (STPA-priv)	Requirements	OP
2017	[1]	Safety	AVs	No	Requirements, New architectures	OP
2017	[39]	Safety, Privacy	E-health	No	Requirements	OP
2017	[16]	Safety	Drones	No	Requirements	OP
2018	[50]	Safety, Security	AVs	Yes (STPA with six-step model)	Requirements	DP, OP
2018	[17]	Safety	RFSs	Yes (STPA with UPPAAL)	Requirements, New architectures	OP
2019	[44]	Safety, Security	Aeronautics	Yes (STPA-sec)	Requirements	OP
2019	[54]	Safety, Security	AVs	No	Requirements, Test cases	OP
2019	[49]	Safety	ASs	No	Requirements, Test cases	OP
2020	[31]	Safety, Security	AVs	Yes (STAMP SnS)	Requirements	DP, OP
2020	[11]	Safety	Cobots	Yes (STPA with Bowtie)	Requirements	OP
2020	[9]	Safety	AMRs	Yes (STPA with FTA)	Requirements	OP
2021	[59]	Safety	ACs	No	Requirements	OP
2021	[38]	Safety	AVs	No	Requirements	OP
2021	[3]	Safety	Cobots	No	Requirements	OP
2021	[32]	Safety	AVs	No	Requirements, Test cases	OP
2021	[27]	Safety	ASs	Yes (STPA with UFoI-E)	Requirements	OP
2021	[4]	Safety	AVs	Yes (SysML-STPA)	Requirements, New architectures	OP
2021	[56]	Safety	AVs	No	Requirements	OP
2021	[19]	Safety	AUVs	Yes (SE-STPA)	Requirements	OP
2021	[10]	Safety	AMRs	Yes (STPA with AHP)	Requirements	OP
2021	[61]	Safety	ASs	Yes (STPA-SynSS)	Requirements	OP
2022	[58]	Safety	ASs	No	Requirements	OP
2022	[25]	Safety	AVs	No	Requirements	OP
2022	[14]	Safety	AMRs, Cobots	No	Requirements, Test cases	OP
2022	[5]	Safety, Reliability	ASs	Yes (STPA with SLIM)	Requirements	OP
2022	[24]	Safety	N/A	Yes (STPA with MRM)	Requirements	OP
2023	[12]	Safety	AMRs	Yes (STPA with SPN)	Requirements	OP
2023	[26]	Safety, Security	AV	No	Requirements, New architectures	OP
2023	[46]	Safety	AV	Yes (STPA with ChatGPT)	Requirements	OP

survey. The general framework is depicted in Fig. 3—for a given LES, grey shaded areas contain new control loops modelled by DeepSTPA while the green region contains the control loops from the original STPA.

Specifically, in Fig. 3, the horizontal axis signifies the ML lifecycle highlighting how data is being processed, while the vertical axis signifies fine-grained functionalities and development activities. The solid line signifies control action (e.g., human commands, messages and data), and the dashed line represents feedback information. Golden boxes highlight activities involve developers. The red dashed box indicates that the ML component is divided into its inner model, showing finer-grained functionalities at lay-wise level.

V. A COMPARATIVE STUDY WITH DEEPSTPA

DeepSTPA is applied on two case studies—object detection (YOLOv3) on AUVs and DRL-based AEB system on AVs. Considering the page limit, we only present the latter, with a baseline in [56] for comparative studies, while cf. our project website[1] for the former.

A. Automatic Emergency Braking Systems

If a collision is about to occur and the driver takes no action or the action is not fast enough, AEB system will automatically initiate braking. AEB is able to detect potential collisions and activate the braking system to slow down the vehicle to avoid the collision or reduce its impact. It is a common subsystem on both conventional and autonomous vehicles. A typical AEB system consists of many components, including signal acquisition, calculation, algorithm, and fusion processes, as well as interfaces with electrical and mechanical parts, sensor systems, and more. Recently ML-based AEB solutions are emerging, e.g., in [15].

[1] https://github.com/YiQi0318/DeepSTPA

B. Baseline Results of STPA with AEB systems

We choose the STPA result of [56] as our baseline, in which an AEB system was analysed. Given the main adaptation of DeepSTPA to STPA is by modelling more comprehensive control loop structures (cf. the grey shaded area compared to the green area in Fig. 3), the analysis will primarily concentrate on comparing the modelling control structures—*if more control actions can be modelled, consequently more UCAs, causal scenarios and safety requirements may also be identified*.

The STPA control loop structure yield from [56], representing a minimum unit AEB system with sensors, controller and actuators, is shown in the green shaded area of Fig. 4 (while omitting those example causal factors leading to hazards in the original baseline paper).

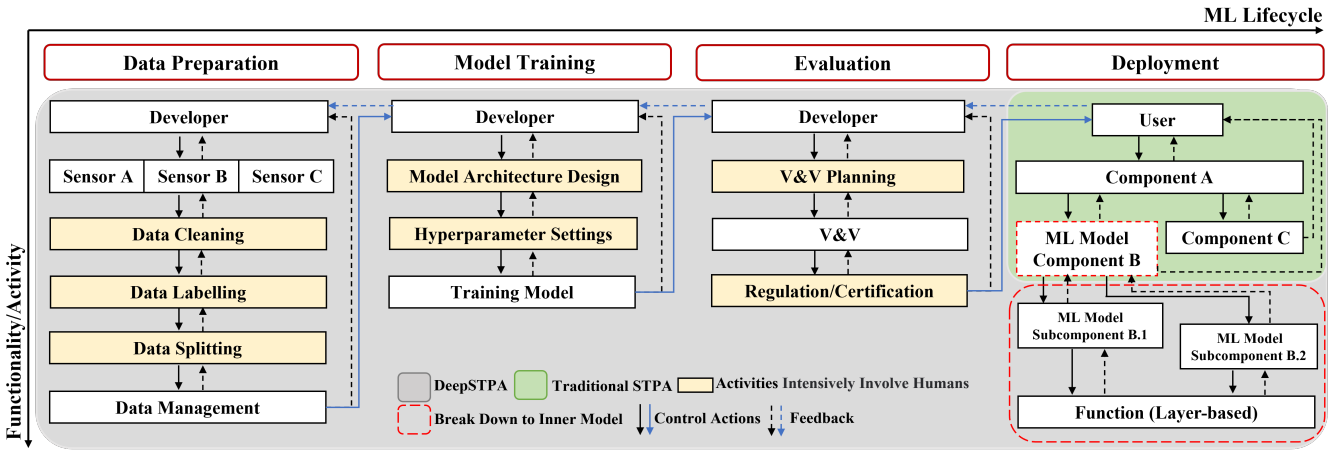

Fig. 3. DeepSTPA control loop structures (grey shaded) in addition to the traditional STPA control loop (green shaded)

C. Results of DeepSTPA applied on DRL-based AEB systems

1) Mishaps Definition: DeepSTPA's first step resembles STPA, where potential mishaps are listed and accidents are determined by LESs' application scenarios. Since AVs colliding with the assets may cause severe monetary loss or human injured, mishaps (**M**) considered are:

- **M1**: Damage to the AVs
- **M2**: Damage to the surrounding assets
- **M3**: Human injured

2) Control Loop Modelling: Fig. 4 shows the complete control loop structure modelled by DeepSTPA for the DRL agent of [15], which is a special case of the general framework in Fig. 3. Specfically, the first stage in the ML lifecycle (i.e., data preparation) is replaced with "environment preparation". This adaptation depends mainly on the type of models used in the ML component. The developer will design the reward function and simulation scenarios to prepare for training the model in the next stage (solid blue line across stages). During the model training stage, it is important to consider the architecture design of the NN, the trade-off between exploration and exploitation (E&E) policy, and configuration of the replay buffer. After obtaining the trained model and setting up the verification and validation (V&V) plan, the V&V steps are carried out in either simulation or real environments, at the evaluation stage. Then, the ML model can be deployed into a LES. The traditional STPA method can be used to analyse potential risks and hazards (green area) at this operation stage. However, DeepSTPA also explores deeper into the functionality/activity dimension. That is, given the core model of the DRL-based AEB is the DQN model, we further develop control loop structures (red dashed box) to represent layer-wise functionalities. Inputs are passed through the hidden layers for feature extraction, parameter reduction, and computation. The output layer selects the appropriate action signal to send to the actuator based on the E&E policy.

3) Unsafe Control Actions: There are 4 parts consisting of a hazardous control action of each UCA, *Source Controller*, *Type* (with the 4 aforementioned categories *T1-T4*), *Control Action* and *Context*. According to [37], the four categories in *Type* are *complete* in terms of defining UCAs. To be more pertinent, DeepSTPA introduces subcategories to indicate new deviations representing ML characteristics applied under the Type *T2*, includeing "Wrong", "Invalid", "Incomplete" and "Perturbed", e.g., "Wrong setting and NN design" is a human error of ML model developers.

In the dimension of ML lifecycle, developers may make human errors in development activities, e.g. (UCA index corresponds to numbers labeled on control actions in Fig. 4):

- **UCA-1** Incomplete reward function design
- **UCA-2** Wrong hyperparameter setting

During the environment preparation stage, the developer is responsible for designing a reward function that effectively communicates the mission goals to the agent (including penalties for collisions or undesired actions). However, human errors can occur, such as neglecting secondary goals that the agent should fulfill. This incomplete reward function design can result in poor performance of the trained model, leading to incorrect action signal outputs from the DRL-based AEB component. These failures can have serious consequences (**M1**, **M2**, and **M3**). UCA-2 is another example highlighting a human error in the model training stage that may affect system-level safety eventually. Note, without modelling control structures from the ML lifecycle dimension, the original STPA cannot identify such UCAs.

If potential risks or hazards are found in a certain component during the deployment stage, DeepSTPA can support detailed safety analysis, tracing back (via blue arrows in Fig. 4) to development stages in the ML lifecycle for that component (**UCA-1** and **UCA-2**). One example is the passing of the reward function and simulation scenarios from the "environment preparation" stage to the "model training" stage, and then placing the trained model in the system after the V&V evaluation stage. This means the ML lifecycle stages formed a control loop structure, allowing us to loop back from the operation process to the ML component development process.

One common UCA related to DRL-based AEB component

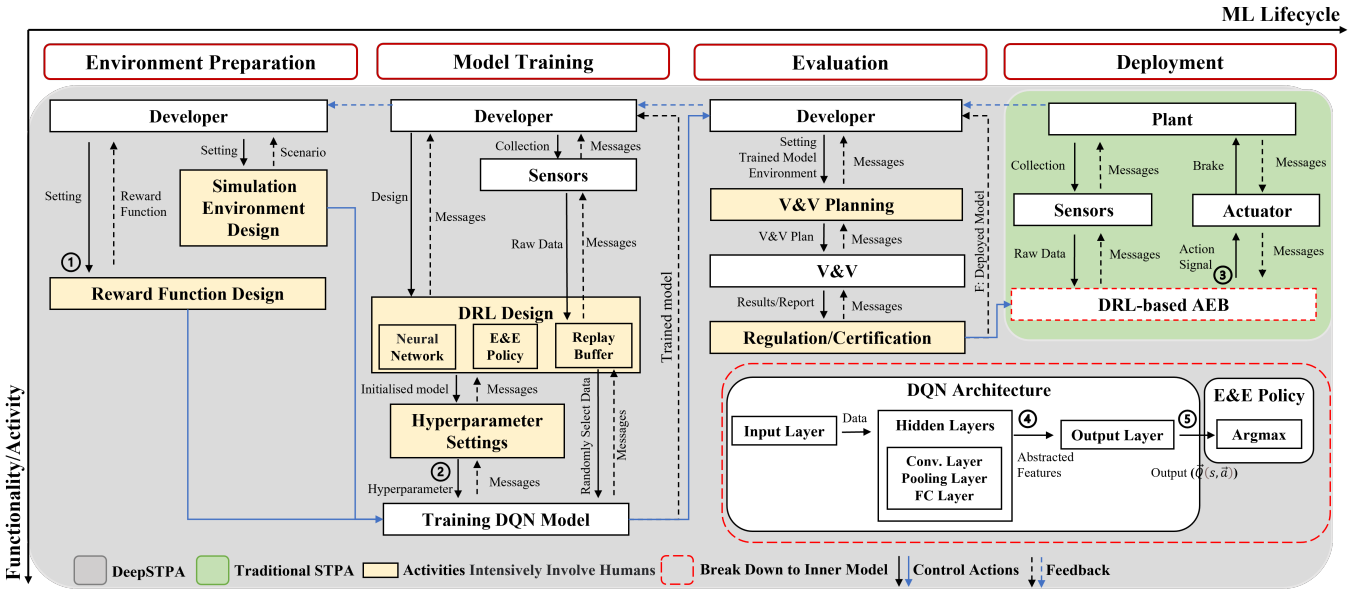

Fig. 4. DeepSTPA control loop structures applied on a DRL-based AEB system

is **UCA-3**: DRL-based AEB component does not provide the correct action signal (*T1*). **UCA-3** can be easily identified using STPA, as it is apparent from the operational control loop structure (green area in Fig. 4). DeepSTPA is an extension of the original STPA that allows for a more detailed analysis that can be further broken down into the internal layers of the ML model. For instance, in the case of a DRL-based AEB system, the core model is the DQN model, and DeepSTPA can help identify that the DQN model does not provide an action signal:

- **UCA-4** The kernel size or stride is too large in the convolutional layer, resulting in a limited amount of information being extracted by the convolutional layer in real-world scenarios[2].
- **UCA-5** There is the gap between the estimated Q value through *argmax* and the actual maximum Q value[3].

UCA-3 can be further broken down to the inner ML model (red dashed box in Fig. 4), which may be due to the DQN model not providing the correct action signal. The analysis revealed that the potential risks and hazards at the layer-wise level, as reflected in **UCA-4** and **UCA-5**, can cause the model to output incorrect action signals, ultimately affecting the entire component's output and system-level safety.

4) Causal Scenarios: In general, STPA considers two causal scenarios (**C**): the reasons behind UCAs and the factors contributing to improper/unexecuted control actions. Referring to the aforementioned **UCA-5**, we present some example causal scenarios (while a more complete list would require DRL experts' knowledge, as expected in STPA [17]):

- **C1**: The model's generalisation ability is inadequate[4].
- **C2**: There exists the explore-exploit dilemma[5].
- **C3**: Sample efficiency is not good[6].
- **C4**: It may involve a sparse reward situation[7].

5) Safety Requirements: From the defined mishaps, UCAs, and causal scenarios, finally a set of safety requirements (**R**) can be derived to enhance the overall safety. One of the common safety requirements involves the creation of safety project checklists or the implementation of consistency checks for specific components. Additionally, standards may be formulated to ensure adherence to safety protocols and guidelines. To address **UCA-5**, effective solutions include:

- **R1**: Apply data augmentation techniques during the model training process[8].
- **R2**: Maintain a balance between exploration and exploitation in reinforcement learning, it can be employed state-of-the-art methods like the ϵ-greedy approach[9].
- **R3**: Incorporate an experience replay[10].
- **R4**: Lead into the reward shaping to relieve the sparse reward situation[11].

These safety requirements help mitigate risks and ensure

[2]Kernel size refers to the window size over the input, and a larger size extracts less information causing performance degradation. Stride determines how many pixels the window should move, and a higher stride can lead to more limited feature extraction.

[3]The *argmax* function selects the maximum value of $\mathbf{Q}(s,\mathbf{a})$ from the NN and outputs its corresponding action signal to the next component.

[4]Generalisation enables good performance on new data beyond training.

[5]It arises explore-exploit dilemma when there is uncertainty about the rewards associated with different actions or strategies.

[6]When the sample efficiency is not good, it means that the model requires a large number of training samples to achieve satisfactory performance.

[7]Sparse rewards refer to instances where the agent receives limited feedback or reward signals from the environment during training.

[8]Data augmentation diversifies and increases training data to improve the model's ability to generalize and handle real-world variations.

[9]The ϵ-greedy approach is a strategy that enables the agent to explore different actions while also exploiting the currently known optimal actions.

[10]Replaying past experiences during training helps the model learn from diverse sets of data, enhancing efficiency and stability.

[11]Reward shaping is a technique that involves adding additional reward signals to guide the RL agent during training. These additional rewards are designed to provide more informative feedback to the agent.

TABLE II
COMPARISON ATTRIBUTE LIST (ADAPTED FROM [56]) AND RESULTS

Attributes	Descriptions	STPA	DeepSTPA
Identify hazards	Comprehensiveness of identified hazards	Partially	Comprehensively
Identify causes	Comprehensiveness of causes of identified hazards	Partially	Comprehensively
Hazard causal factors	Type of identified hazard causal factors	Hardware, Software	Hardware, Software, ML Model
Life-cycle phase	Method can be applied in which phase of the product life-cycle	Operation phase	Development phase, Operation phase
Time and cost	A time and cost required for safety analysis with the method	Low	High
Complexity/difficulty	Relative complexity/difficulty of the method	Low	High

that safety considerations are effectively incorporated into the operation process.

In summary, DeepSTPA follows the same methodology as STPA—it retains the five basic steps, but enhances the 2nd step significantly by considering layer-wise functionalities inside ML models and development activities spanning the ML lifecycle (cf. the grey shaded area in Fig. 3). Thanks to such more comprehensive modelling of control loop structures, DeepSTPA can provide causal scenarios and safety requirements for ML specific UCAs that complements STPA.

D. Comparison Methodology and Results

We reuse the comparison methodology from [56] (designed for a diverse range of safety analysis methods), but reduce the set of attributes to suit our specific goal on comparing DeepSTPA to STPA. Specifically, we assess and compare DeepSTPA and STPA according to the attributes list in Table II. The six attributes collectively encompasses two aspects: *analysis results* comparison and *analysis process* comparison. In Table II, the first three rows correspond to the analysis results comparison, while the remaining rows pertain to the analysis process comparison.

The summarised resutls of the comparative study between STPA and DeepSTPA are presented in Table II. The table reveals that only a part of the hazards can be identified in STPA, while DeepSTPA provides a more comprehensive analysis results (hazards, causes and mitigations). This is non-surprising, given that DeepSTPA can model more control actions (during both the development process and fine-grained ML functionalities in the operation) and consequently more UCAs, causal scenarios and safety requirements can be derived. However, the utilisation of DeepSTPA comes at the cost of increased complexity and longer time requirements compared to STPA.

VI. RELATED WORK

STPA was introduced as a replacement for traditional safety analysis methods, e.g., HAZOP [35], in the context of handling complex systems [21]. Researchers explored the combination of STPA with other techniques and its adaptation to a broad range of domains. Rodriguez and Diaz [48] study the integration of STPA with functional models for process hazard analysis, while Kondo et al. [33] adapt the STAMP framework by modifying its terminology to incorporate risk analysis for industrial control systems and cybersecurity. Pasman's work [42] offers a systematic approach to risk control and explores new and improved processes, as well as various risks and resilience analysis tools. There have been surveys conducted on STPA. Harkleroad et al. [28] review the general properties of STPA, summarising that STPA views risk assessment as a top-down control issue instead of some commonly used bottom-up component reliability approaches. Patriarca et al. [43] have recently summarised the history and current status of STAMP, which encompasses STPA as one of its related techniques, highlighting the application areas of surveyed papers. To the best of our knowledge, none of the existing surveys have specifically focused on LESs like ours. Driven by the same motivation as DeepSTPA (i.e., a new way of doing safety analysis for LESs), in [45], authors develop a new method by featuring HAZOP a hierarchical structure, while we are extending STPA which is arguably more effective for complex systems like LESs.

VII. CONCLUSIONS

In this paper, we conduct a survey of 31 papers selected from recent literature of applying STPA to LESs. Based on the survey results, gaps in the state-of-the-art are identified for which we propose a new practice, DeepSTPA, adapting STPA from the dimensions of ML development process and fine-grained functionalities of layers inside ML models. As demonstrated by the comparative case study on AVs, we conclude that DeepSTPA is more effective than STPA when applied to LESs, which can be reflected through the more comprehensive safety analysis results.

REFERENCES

[1] A. Abdulkhaleq, D. Lammering, S. Wagner, J. Röder, N. Balbierer, L. Ramsauer, T. Raste, and H. Boehmert, "A systematic approach based on STPA for developing a dependable architecture for fully automated driving vehicles," *Procedia Eng*, vol. 179, pp. 41–51, 2017.

[2] A. Abdulkhaleq and S. Wagner, "XSTAMPP: an extensible STAMP platform as tool support for safety engineering," *STAMP WS*, 2015.

[3] A. Adriaensen, L. Pintelon, F. Costantino, G. D. Gravio, and R. Patriarca, "An STPA safety analysis case study of a collaborative robot application," *IFAC*, vol. 54, no. 1, pp. 534–539, 2021.

[4] A. Ahlbrecht and O. Bertram, "Evaluating system architecture safety in early phases of development with MBSE and STPA," in *ISSE*, 2021.

[5] S. I. Ahn, R. E. Kurt, and O. Turan, "The hybrid method combined STPA and SLIM to assess the reliability of the human interaction system to the emergency shutdown system of lng ship-to-ship bunkering," *Ocean Engineering*, vol. 265, p. 112643, 2022.

[6] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[7] E. Asaadi, E. Denney, and G. Pai, "Quantifying assurance in learning-enabled systems," in *SafeComp'20*, ser. LNCS, vol. 12234, 2020.

[8] R. Ashmore, R. Calinescu, and C. Paterson, "Assuring the machine learning lifecycle: Desiderata, methods, and challenges," *ACM Comput. Surv.*, vol. 54, no. 5, 2021.

[9] C. Bensaci, Y. Zennir, and D. Pomorski, "A new approach to system safety of human-multi-robot mobile system control with STPA and FTA," *Algerian Journal of Signals and Systems*, pp. 79–85, 2020.

[10] C. Bensaci, Y. Zennir, D. Pomorski, F. Innal, and Y. Liu, "Distributed vs. hybrid control architecture using STPA and AHP - application to an autonomous mobile multi-robot system," *Int. Journal of Safety and Security Engin.*, vol. 11, pp. 1–12, 3 2021.

[11] C. Bensaci, Y. Zennir, D. Pomorski, F. Innal, Y. Liu, and C. Tolba, "STPA and Bowtie risk analysis study for centralized and hierarchical control architectures comparison," *Alexandria Engineering Journal*, vol. 59, no. 5, pp. 3799–3816, 2020.

[12] C. Bensaci, Y. Zennir, D. Pomorski, F. Innal, and M. A. Lundteigen, "Collision hazard modeling and analysis in a multi-mobile robots system transportation task with STPA and SPN," *Reliability Engineering & System Safety*, vol. 234, p. 109138, 2023.

[13] R. Bloomfield, H. Khlaaf, P. R. Conmy, and G. Fletcher, "Disruptive innovations and disruptive assurance: Assuring machine learning and autonomy," *Computer*, 2019.

[14] L. Buysse, D. Vanoost, J. Vankeirsbilck, J. Boydens, and D. Pissoort, "Case study analysis of STPA as basis for dynamic safety assurance of autonomous systems," in *EDCC*, 2022.

[15] H. Chae, C. M. Kang, B. Kim, J. Kim, C. C. Chung, and J. W. Choi, "Autonomous braking system via deep reinforcement learning," in *IEEE 20th Int. Conf. on Intelligent Transportation Systems*, 2017.

[16] M. M. Chatzimichailidou, N. Karanikas, and A. Plioutsias, "Application of STPA on small drone operations: A benchmarking approach," *Procedia Engineering*, 2017, 4th European STAMP Workshop.

[17] A. L. Dakwat and E. Villani, "System safety assessment based on STPA and model checking," *Safety Science*, vol. 109, 2018.

[18] Department for Digital, Culture, Media and Sport, U.K., "Establishing a pro-innovation approach to regulating AI," Tech. Rep., 2022.

[19] D. Dghaym, T. S. Hoang, S. R. Turnock, M. Butler, J. Downes, and B. Pritchard, "An STPA-based formal composition framework for trustworthy autonomous maritime systems," *Safety Science*, 2021.

[20] Y. Dong, W. Huang, V. Bharti, V. Cox, A. Banks, S. Wang, X. Zhao, S. Schewe, and X. Huang, "Reliability Assessment and Safety Arguments for Machine Learning Components in System Assurance," *ACM Trans. Embed. Comput. Syst.*, vol. 22, no. 3, 2023.

[21] S. R. e Silva, "System theoretic process analysis: A literature survey on the approaches used for improving the safety in complex systems," in *Information Systems for Industry 4.0*, 2019, pp. 97–114.

[22] L. Edwards, "Regulating ai in europe: four problems and four solutions," *Retrieved March*, vol. 15, 2022.

[23] S. Friese, "User's manual for ATLAS. ti 6.0, ATLAS. ti scientific software development," *Berlin: GmbH*, 2013.

[24] S. Gautham, G. Bakirtzis, A. Will, A. V. Jayakumar, and C. R. Elks, "STPA-driven multilevel runtime monitoring for In-time hazard detection," in *SafeComp'20*, 2022, pp. 158–172.

[25] G. Ge, L. Sun, and Y. F. Li, "A systematic approach to develop an autopilot sensor monitoring system for autonomous delivery vehicles based on the STPA method," in *ISSREW*, 2022, pp. 318–325.

[26] S. Ghosh, A. Zaboli, J. Hong, and J. Kwon, "An integrated approach of threat analysis for autonomous vehicles perception system," *IEEE Access*, vol. 11, pp. 14 752–14 777, 2023.

[27] N. H. C. Guzman, J. Zhang, J. Xie, and J. A. Glomsrud, "A comparative study of STPA-extension and the UFoI-E method for safety and security co-analysis," *Reliability Engineering & System Safety*, 2021.

[28] E. Harkleroad, A. Vela, and J. Kuchar, "Review of systems-theoretic process analysis (STPA) method and results to support NextGen concept assessment and validation," *Proj. Report: ATC-427 MIT*, 2013.

[29] R. Hawkins, C. Paterson, C. Picardi, Y. Jia, R. Calinescu, and I. Habli, "Guidance on the assurance of machine learning in autonomous systems (AMLAS)," *arXiv preprint arXiv:2102.01564*, 2021.

[30] X. Huang, G. Jin, and W. Ruan, *Machine Learning Safety*. Springer, 2023.

[31] T. Kaneko, N. Yoshioka, and R. Sasaki, "STAMP S&S: Safety & security scenario for specification and standard in the society of AI/IoT," in *QRS-C*, 2020, pp. 168–175.

[32] S. Khastgir, S. Brewerton, J. Thomas, and P. Jennings, "Systems approach to creating test scenarios for automated driving systems," *Reliability Engineering & System Safety*, vol. 215, p. 107610, 2021.

[33] S. Kondo, H. Sakashita, S. Sato, T. Hamaguchi, and Y. Hashimoto, "An application of STAMP to safety and cyber security for ICS," in *Computer Aided Chemical Engineering*, 2018, vol. 44, pp. 2335–2340.

[34] P. Koopman, A. Kane, and J. Black, "Credible autonomy safety argumentation," in *27th Safety-Critical Systems Symp.*, 2019.

[35] H. Lawley, "Operability studies and hazard analysis," *Chem. Eng. Prog.*, vol. 70, no. 4, 1974.

[36] N. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*, ser. Engineering systems. MIT Press, 2011.

[37] N. G. Leveson and J. P. Thomas, "STPA handbook," *USA*, 2018.

[38] B. Li, S. Shang, and Y. Fu, "The application of stpa in the development of autonomous vehicle functional safety," in *ICAA*, 2021, pp. 863–868.

[39] K. Mindermann, F. Riedel, A. Abdulkhaleq, C. Stach, and S. Wagner, "Exploratory study of the privacy extension for system theoretic process analysis (stpa-priv) to elicit privacy risks in ehealth," in *IEEE 25th Int. Req. Engi. Conf. workshops (REW)*. IEEE, 2017, pp. 90–96.

[40] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[41] G. Moreira, D. R. Pleffken, C. Cerqueira, and W. Santos, "STPA analysis over the earlier phases of brazilian aerospace products life cycle using OPM," in *ICMAE*, 2022, pp. 465–471.

[42] H. J. Pasman, *Risk analysis and control for industrial processes-gas, oil and chemicals: a system perspective for assessing and avoiding low-probability, high-consequence events*, 2015.

[43] R. Patriarca, M. Chatzimichailidou, N. Karanikas, and G. Di Gravio, "The past and present of system-theoretic accident model and processes (STAMP) and its associated techniques: A scoping review," *Safety Science*, vol. 146, p. 105566, 2022.

[44] D. P. Pereira, C. Hirata, and S. Nadjm-Tehrani, "A STAMP-based ontology approach to support safety and security analyses," *Journal of Information Security and Applications*, vol. 47, pp. 302–319, 2019.

[45] Y. Qi, P. M. Ryan, W. Huang, X. Zhao, and X. Huang, "A Hierarchical HAZOP-Like Safety Analysis for Learning-Enabled Systems," in *AI Safety at IJCAI-22*, vol. 3215. Vienna, Austria: CEUR, 2022, p. 10.

[46] Y. Qi, X. Zhao, and X. Huang, "Safety analysis in the era of large language models: A case study of STPA using ChatGPT," *arXiv 2304.01246*, 2023.

[47] M. Rejzek and C. Hilbes, "Use of STPA as a diverse analysis method for optimization and design verification of digital instrumentation and control systems in nuclear power plants," *NED*, pp. 125–135, 2018.

[48] M. Rodríguez and I. Díaz, "A new functional systems theory based methodology for process hazards analysis," *CACE*, pp. 703–708, 2014.

[49] Rokseth, Børge, Haugen, Odd Ivar, and Utne, Ingrid Bouwer, "Safety verification for autonomous ships," *MATEC Web Conf.*, 2019.

[50] G. Sabaliauskaite, L. Liew, and J. Cui, "Integrating autonomous vehicle safety and security analysis using STPA method and the six-step model," *Int. J. on Adv. in Security*, vol. 11, pp. 160–169, 2018.

[51] M. L. Salgado and M. S. de Sousa, "Cybersecurity in aviation: the STPA-sec method applied to the TCAS security," in *LADC*, 2021.

[52] S. S. Sandha, *Learning-enabled Cyber-Physical Systems: Challenges and Strategies*. University of California, Los Angeles, 2022.

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

[54] S. Sharma, A. Flores, C. Hobbs, J. Stafford, and S. Fischmeister, "Safety and Security Analysis of AEB for L4 Autonomous Vehicle Using STPA," in *Workshop on ASD*, vol. 68, 2019, pp. 5:1–5:13.

[55] S. Sultana, P. Okoh, S. Haugen, and J. E. Vinnem, "Hazard analysis: Application of stpa to ship-to-ship transfer of lng," *JLPPI*, 2019.

[56] L. Sun, Y.-F. Li, and E. Zio, "Comparison of the HAZOP, FMEA, FRAM, and STPA Methods for the Hazard Analysis of Automatic Emergency Brake Systems," *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, vol. 8, no. 3, 2021.

[57] A. D. Williams, "System theoretic process analysis (STPA): Overview of sandia uses to address national security problems." 2019.

[58] T. Yamada, M. Sato, R. Kuranobu, R. Watanabe, H. Itoh, M. Shiokari, and T. Yuzui, "Evaluation of effectiveness of the STAMP / STPA in risk analysis of autonomous ship systems," *JPCS*, 2022.

[59] W. Zhang, X. Meng, J. Wang, T. Li, Q. Shan, and F. Teng, "Safety analysis of automatic crane trolley running system based on STAMP/STPA," in *ICCSS*, 2021.

[60] X. Zhao, A. Banks, J. Sharp, V. Robu, D. Flynn, M. Fisher, and X. Huang, "A Safety Framework for Critical Systems Utilising Deep Neural Networks," in *SafeComp'20*, ser. LNCS, vol. 12234, 2020.

[61] X. Zhou, Z. Liu, F. Wang, and Z. Wu, "A system-theoretic approach to safety and security co-analysis of autonomous ships," *Ocean Engineering*, vol. 222, p. 108569, 2021.