ReSonAte: A Runtime Risk Assessment Framework for Autonomous Systems

Charles Hartsell§, Shreyas Ramakrishna§, Abhishek Dubey, Daniel Stojcsics, Nagabhushan Mahadevan, and Gabor Karsai
Institute for Software Integrated Systems, Vanderbilt University

Abstract—Autonomous Cyber Physical Systems (CPSs) are often required to handle uncertainties and self-manage the system operation in response to problems and increasing risk in the operating paradigm. This risk may arise due to distribution shifts, environmental context, or failure of software or hardware components. Traditional techniques for risk assessment focus on design-time techniques such as hazard analysis, risk reduction, and assurance cases among others. However, these static, design-time techniques do not consider the dynamic contexts and failures the systems face at runtime. We hypothesize that this requires a dynamic assurance approach that computes the likelihood of unsafe conditions or system failures considering the safety requirements, assumptions made at design time, past failures in a given operating context, and the likelihood of system component failures. We introduce the ReSonAte dynamic risk estimation framework for autonomous systems. ReSonAte reasons over Bow-Tie Diagrams (BTDs) which capture information about hazard propagation paths and control strategies. Our innovation is the extension of the BTD formalism with attributes for modeling the conditional relationships with the state of the system and environment. We also describe a technique for estimating these conditional relationships and equations for estimating risk based on the state of the system and environment. To help with this process, we provide a scenario modeling procedure that can use the prior distributions of the scenes and threat conditions to generate the data required for estimating the conditional relationships. To improve scalability and reduce the amount of data required, this process considers each control strategy in isolation and composes several single-variate distributions into one complete multi-variate distribution for the control strategy in question. Lastly, we describe the effectiveness of our approach using two separate autonomous system simulations: CARLA and an unmanned underwater vehicle.

Index Terms—System Risk Management, Dynamic Risk, Assurance Case, Hazard Analysis, Bow-Tie Diagram

I. INTRODUCTION

Autonomous Cyber Physical Systems (CPSs) are expected to handle uncertainties and self-manage the system operation in response to problems and increase in risk to system safety. This risk may arise due to distribution shifts, environmental context or failure of software or hardware components. System Risk Management (SRM) has been a well-known approach used to assess the system’s operational risk. It involves design-time activities such as hazard analysis for identifying the system’s potential hazards, risk assessment to identify the risk associated with the identified hazards, and a system-level assurance case to argue the system’s safety. However, the design-time hazard analysis and risk assessment information it uses is inadequate in highly dynamic situations at runtime. To better address the dynamic operating nature of CPSs, dynamic assurance approaches such as runtime certification, dynamic assurance cases, and modular safety certificates have been proposed. These approaches extend the assurance case to include system monitors whose values are used to update the reasoning strategy at runtime. A prominent approach for using the runtime information has been to design a discrete state space model for the system, identity the risk associated with each possible state transition action, then perform the action with the least risk.

The effectiveness of dynamic risk estimation has been demonstrated in the avionics and medical domains, but these techniques often encounter challenges with state space explosion which may limit their applicability to relatively low-complexity systems. Also, real-time CPSs often have strict timing deadlines in the order of tens of milliseconds requiring any dynamic risk assessment technique to be computationally lightweight. Besides, the dynamic risk assessment technique should also take into consideration the uncertainty introduced by the Learning Enabled Components (LECs) because of Out-Of-Distribution (OOD) data. Assurance monitors, Assurance monitors, are a type of OOD detector often used to tackle the OOD data problem. The output of these monitors should be considered when computing the dynamic risk.

Recently, there is a growing interest in dynamic risk assessment of autonomous CPS. For example, the authors in have used Dynamic Bayesian Networks to incorporate the broader effect of spatio-temporal risk gathered from road information on the system’s operational risk. To perform dynamic risk assessment of a system with autonomous components, this paper introduces the Runtime Safety Evaluation in Autonomous Systems (ReSonAte) framework. ReSonAte uses the design-time hazard analysis information to build Bow-Tie Diagrams (BTDs) which describe potential hazards to the system and how common events may escalate to consequences due to those hazards. The risk posed by these hazards can change dynamically since the frequency of events and effectiveness of hazard controls may change based on the state of the system and environment. To account for these dynamic events at runtime, ReSonAte uses design-time BTD models along with information about the system’s current state derived from system monitors (e.g. anomaly detectors, assurance monitors, etc.) and the operating environment (e.g. weather, traffic, etc.) to estimate dynamic hazard rates. The estimated hazard rate can be used for high-level decision
making tasks at runtime, to support self-adaptation of CPSs.

The specific contributions of this paper are the following. We present the ReSonAte framework and outline the dynamic risk estimation technique which involves design-time measurement of the conditional relationships between hazard rates and the state of the system and environment. These conditional relationships are then used at run-time along with state observations from multiple sources to dynamically estimate system risk. Further, we describe a process that uses an extended BTD model for estimating the conditional relationships between the effectiveness of hazard control strategies and the state of the system and environment. To improve scalability and reduce the amount of data required, this process considers each control strategy in isolation and composes several single-variate distributions into one complete multivariate distribution for the control strategy in question. A key contribution is our scenario specific language that enables data collection by specifying prior distributions over threat conditions and environmental conditions and initial conditions of the vehicle. We implement ReSonAte for an Autonomous Vehicle (AV) example in the CARLA simulator [14] and through comprehensive simulations across 600 executions, we show that there is a strong correlation between our risk estimates and the state of the system and environment. We define risk as a comprehensive simulation across 600 executions, we show that there is a strong correlation between our risk estimates and the state of the system and environment. To improve scalability and reduce the amount of data required, this process considers each control strategy in isolation and composes several single-variate distributions into one complete multivariate distribution for the control strategy in question. A key contribution is our scenario specific language that enables data collection by specifying prior distributions over threat conditions and environmental conditions and initial conditions of the vehicle. We implement ReSonAte for an Autonomous Vehicle (AV) example in the CARLA simulator [14] and through comprehensive simulations across 600 executions, we show that there is a strong correlation between our risk estimates and the state of the system and environment. We define risk as a

II. RELATED WORK

System health management [15], [16] with model based reasoning have been popularly used for self-adaptation of traditional CPSs. As discussed in [17], a pre-requisite in these approaches has been that the system has knowledge about itself, its objectives, and its operating environments, including the ability to estimate when the adaption should occur. One way to do this is to estimate the operational risk to the system at runtime. Estimating the runtime risk is more difficult in autonomous systems with LECs due to the black box nature of the learning components and their susceptibility to distribution shifts and environmental changes.

Our hypothesis is that we can use BTDs derived from the system information and static assurance cases and use them to compose the anomaly and the threat likelihoods at runtime, including the likelihood for the failure of LECs [1], [11], [12], [18]. BTDs are graphical models that provide a mechanism to learn the conditional relationships between threat events, hazards, and the success probability of barriers. BTDs have been proactively used for qualitative risk assessment at design-time [19], [20], and has been recently combined with assurance arguments for operational safety assurance [21], [22].

Although BTDs have been proactively used for risk assessment, there are several limitations in using them for quantitative risk computations, they are: (1) Static structure - BTDs have mostly been used as a graphical visualization tool for hazard analysis, and its static structure limits real-data updating which is required for dynamic risk estimation [23]. (2) Reliance on domain experts - quantitative risk estimations mostly rely on domain experts to compute the probabilities for the BTD events such as threats, and barriers [24]. (3) Data uncertainty - introduced because of data non-availability or insufficiency, and expert’s limited knowledge makes it difficult to compute quality probability estimates [25].

Several approaches have been proposed to overcome these limitations and make BTDs suitable for quantitative risk assessment. Bayesian techniques are one widely used approach that dynamically learns the BTD structure from design-time data, and updates the conditional probabilities for its events (e.g., threats, barriers) [26], [27]. Further, [28] transforms a BTD into a Bayesian Network where each node of the BTD is modelled as a Bayesian Node. The other approach uses techniques such as fuzzy set and evidence theory to address the data uncertainty problems [29], [29]. In this approach, the expert’s knowledge is translated into numerical quantities that are used in the BTDs. [30] extends BTDs with petri-nets models and monte-carlo simulations to generate data.

Although prior approaches have made BTDs suitable for quantitative risk estimation, the design-time hazard information used for risk estimation is inadequate for CPSs. The main focus of this work is to dynamically estimate risk by fusing design-time information captured in the BTDs with runtime information about the system and the environment. Additionally, we concentrate on generating large-scale simulation data for conditional probability estimation of the BTD events.

III. ReSonAte Framework

The goal of the ReSonAte framework is the dynamic estimation of risk based on runtime observations about the state of the system and environment. We define risk as a product of the likelihood and severity of undesirable events, or consequences. Fig. 1 outlines the ReSonAte workflow which is divided into design-time and run-time steps. The design-time steps include System Analysis, Hazard Analysis, Assurance Case Construction, and BTD Modeling. Sections III-A through III-B provide background information about each of these well-studied techniques. However, our BTD formalism described in Section III-B is distinct from existing formalisms with additional model attributes and restrictions for the ReSonAte framework. Sections III-C through III-E introduce the risk calculation equations, the conditional probability estimation process, and the dynamic assurance case evaluation method of the ReSonAte framework respectively.

A. Background

1) System Analysis: System analysis involves the design-time analysis of the system operation, system faults, the available runtime monitors, and the operating environment such as weather (e.g. rain, snow, fog, etc.), traffic, etc. Thereafter, we perform failure mode analysis to identify the possible component faults and their potential impacts on the safety of the system. Fault propagation paths can be described with the use of an appropriate fault modeling language (e.g. Timed Failure
Propagation Graph (TFPG) \({\text{[31]}}\), and run-time monitors for fault identification can be designed as appropriate. Monitors to detect faults in traditional components have been designed in prior work \({\text{[32]}}\), but monitors for LECs are often more complex (e.g. assurance monitors \({\text{[11]}}\)). Alarms raised by these monitors at run-time can then be fed into an appropriate fault diagnosis engine (e.g. the TFPG reasoning engine) to isolate particular fault modes that may be present in the system.

2) Hazard Analysis: Hazard analysis involves the identification of events that may lead to hazardous conditions, implementation of barriers to prevent loss of control over hazard conditions or recovery control after a loss, and estimation of the risk posed by the identified hazards. Guidelines for performing hazard analysis are available in a variety of domains including the ISO 26262 standard \({\text{[33]}}\) for the automotive domain and the FAA System Safety Handbook \({\text{[34]}}\) for the aerospace domain.

3) Assurance Case Construction: An assurance case \({\text{[4]}}\) is a structured argument supported by a body of evidence which shows that system goals will be met in a defined operating environment and is often documented using the Goal Structuring Notation (GSN) \({\text{[35]}}\). Multiple authors have noted the importance of having an hazard analysis and risk reduction arguments within an assurance case \({\text{[36], [37]}}\). In this work, we only concentrate on the hazard analysis argument (Fig. 1) of the assurance case. Prior works \({\text{[6], [38]}}\) can be referred for a complete dynamic assurance case.

B. Our Contributions: Bow-Tie Formalization & Extensions

To dynamically estimate risk the hazard models must contain information about the expected rate of threat events and the success probability of barriers, both of which may be conditional on the current conditions. In this section, we present a BTD formalization which includes this conditional information not captured in existing BTD interpretations. Under our formalism, each node has an associated function that is conditional on the state of the system and environment. Before constructing a BTD, we assume the following sets have been identified: all potential hazard classes \(H\), all events \(E\) relevant for hazard analysis, barriers \(B\) which may either prevent a loss of control or recover after such a loss has occurred, possible system failure modes \(FM\), and event severity classes \(SV\). We also assume a function \(f_a\) which maps each severity class to the maximum acceptable rate of occurrence threshold \(f_a : SV \rightarrow \mathbb{R}\) has been defined.

For risk calculation, we define the state of the system and the environment as \(S = (F, e, m)\) where:

- \(F\) is a set of failure modes currently present in the system. \(F \subseteq FM\)
- \(e\) is an \(n\)-tuple describing the current environmental conditions where \(n\) is the number of environmental parameters that are relevant for risk calculations and each parameter may be continuous or discrete valued.
- \(m\) is a \(k\)-tuple containing the outputs of runtime monitors in the system where \(k\) is the number of such monitors and each monitor may be continuous or discrete valued.

When modeling the environment, the system developer should choose an appropriate level of abstraction based on which environmental parameters are relevant for risk calculation. More environmental parameters may increase the fidelity of the model but also increases the required effort when determining conditional probabilities for events and barriers. It is not required to consider every environmental parameter for all events and barriers. Instead, each event or barrier may be conditional upon a smaller subset of environmental parameters that have the largest impact on that particular event or barrier. Formally, we define a Bow-Tie Diagram as a tuple \((N, C, f_i, h, f_d, f_t, f_b, f_e)\) where:

- \(N\) is a set of nodes where each node represents either an event \(e \in E\) or a barrier \(b \in B\).
Fig. 2: Example Bow-Tie Diagram for autonomous vehicle example "Roadway Obstruction" hazard created using the ALC Toolchain. Each block includes a brief description and the type of each node is denoted at the top of the block. The equations depicted are described in Section III-B.

- $C$ represents a set of directed connections such that $C \subseteq N \times N$. For a given connection $c \in C$, $src(c)$ and $dst(c)$ represent the source and destination of $c$ respectively.
- The tuple $(N, C)$ is a Directed Acyclic Graph (DAG) representing the temporal ordering of events as well as the barriers which may break this ordering and prevent further propagation of events.
- The function $f_s$ gives the type of each node. $f_s : N \rightarrow \{\text{event, barrier}\}$. Using this function, we let $N_e = \{n \in N \mid f_s(n) = \text{event}\}$ and $N_b = \{n \in N \mid f_s(n) = \text{barrier}\}$. This gives the following properties: $N_c \subseteq E$, $N_b \subseteq B$, $N = N_e \cup N_b$, and $N_e \cap N_b = \emptyset$.
- $h \in H$ is the hazard class associated with the BTD.
- The function $f_d$ gives a textual description of each node. $f_d : N \rightarrow \text{string}$
- $f_b$ is a probability function conditional on the state of the system and environment defined for each barrier node. This function represents the probability that the barrier will successfully prevent further event propagation. $f_b : N_b \times S \rightarrow [0, 1]$.
- $f_t$ is a function conditional on the state of the system and environment defined for each event node. It gives the expected frequency of the event over a fixed period. The values of this function must be specified for threat events (i.e. root events with no preceding input events), but can be calculated for subsequent events in the diagram as discussed in Section III-C. $f_c : N_c \times S \rightarrow [0, \infty]$.
- A function $f_s$ which maps each event to the appropriate severity class. $f_s : N_c \rightarrow SV$

BTDs are often constructed in a chained manner where a consequence event in one section of the BTD may serve as a threat or top event in a subsequent section of the same BTD. However, such a chained BTD can be broken into multiple, single-scope BTDs where each event has one unique type (i.e. threat, top event, or consequence). As a simplification, ReSonAte operates only on these single-scope BTDs which satisfy the additional restrictions listed below. Note that the symbol $\Rightarrow$ is used to denote precedence in the graph. That is, given two nodes $a, b \in N$, then $a \Rightarrow b$ states that $a$ precedes $b$ in the BTD, but this does not necessitate a direct edge such that $a \rightarrow b$. Instead, there may be any number of intermediate nodes $c_i \in N$ such that $a \rightarrow c_1 \rightarrow c_2 \rightarrow \ldots \rightarrow c_n \rightarrow b$.
- Exactly one event must be designated as the top event, denoted $e_{top}$.
- There must be at least one threat, i.e. $\exists t \in N_t \mid t \Rightarrow e_{top}$ and $n \in N_c \mid n \Rightarrow t$. We denote the set of all threat events as $N_t$.
- There must be at least one consequence, i.e. $\exists c \in N_c \mid e_{top} \Rightarrow c \wedge \exists n \in N_c \mid n \Rightarrow c$. We denote the set of all threat events as $N_c$.
- All events must be a threat, a top event, or a consequence, i.e. No intermediate events.
- All barriers must lie between a threat and the top event, or between the top event and a consequence. i.e. $\forall b \in N_b$ either $t \Rightarrow b \Rightarrow e_{top}$ or $e_{top} \Rightarrow b \Rightarrow c$.
- No branching or joining of the graph is allowed, except for at the top event.

For the example BTD shown in Fig. 2 the type of each event is denoted on the top of the block and we can define the sets $N_c = \{T1, T2, TOP, C1\}$ and $N_b = \{B1, B2, B3\}$. For each event $e \in N_c$, the associated severity class is given by the function $f_s(e)$ which is shown under each event block. Each of the threats, $T1$ and $T2$, have been assigned to the “None” severity class because these events are considered to be common occurrences that do not result in any safety violation by themselves. The top event $TOP$ has been assigned to the “Minor” severity class since this event can still be mitigated before a safety violation occurs but mitigation requires the Automatic Emergency Braking System (AEBs) to override the primary LEC controller. Finally, the consequence $C1$ has been assigned to the “Catastrophic” severity class since this event is a safety violation and may result in significant damage to the system or environment.

The conditional functions $f_c$ and $f_b$ are shown near their respective nodes in Fig. 2. The functions $f_c(T1, s)$ and $f_c(T2, s)$ give the expected frequency of threats $T1$ and $T2$ in units of expected number of occurrences per minute, and their values were measured for our simulator configuration described in Section IV. The probability function for barriers $B1$ and $B2$, $f_b(x = (B1, B2), s)$, shows how these barriers are dependent on...
both the continuous-valued output from the assurance monitor and on the binary state of other monitors. A sigmoid function, \( P(x|s.m.LEC) \), is used to capture the conditional relationship with the assurance monitor output. Finally, \( f_s(B,3,s) \) shows how barrier \( B \) is less likely to succeed as the precipitation increases and will not function in the case of a radar failure. Section [III-C] explains the generic process for conditional probability estimation and Section [IV-A] provides more detail on the functions in this BTD.

C. Run-time Risk Computation

Each BTD includes functions describing the conditional frequency for all threat events and the conditional probability of success for all barrier nodes. However, the likelihood of each consequence is necessary to estimate the overall level of risk for the system. These probabilities can be calculated by the propagation of the initial threat rates through the BTD. When a particular event occurs, barrier nodes reduce the probability that the event will continue to propagate through the BTD based on the following equation:

\[
R(e_2|s) = R(e_1|s)P(b_1 \land b_2 \land ... \land b_n|s)
\]

where \( e_1, e_2 \in N_c \) and \( b_1 \in N_b \) such that \( e_1 \rightarrow b_1 \rightarrow b_2 \rightarrow ... \rightarrow b_n \rightarrow e_2 \). \( R(e_i|s) \) represents the frequency of event \( e_i \) given state \( s \). Eq. (1) makes the assumption that no barriers share any common failure modes and that the effectiveness of each barrier is independent from the outcome of other barriers. Similarly, \( P(b_i|s) \) represents the probability that barrier \( b_i \) will fail to prevent event propagation given state \( s \), i.e. \( P(b_i|s) = 1 - P(b_i|s) \). Letting \( S \) represent the set of all states we are concerned with, then we can calculate the overall frequency of \( e_2 \) as \( R(e_2) = \sum_{s \in S} R(e_2|s)P(s) \) where \( P(s) \) is the probability of each particular state \( s \in S \). As discussed in the BTD formalization in Section [III-B], no joining or splitting of paths is allowed in a BTD with the exception of the top event \( e_{top} \). We treat the top event as a summation operation for all incoming edges, i.e. any threat event may independently cause a top event if the associated barriers are unsuccessful. All outgoing edges from the top event are treated as independent causal chains, i.e. any potential consequence may occur from a top event, independent of other consequences in the BTD. The probability for the top event can be calculated as:

\[
R(t_{top}) = \sum_{s \in S} [R(t|s)P(s)[\prod_{i=1}^{n} P(b_i|s)]]
\]

where \( R(e_{top}) \) is the frequency for the top event and \( P(t_{top}) \) represents the contribution of each threat \( t \) to this rate after passing through any intermediate prevention barriers \( b_i \in N_b \) such that \( t_1 \rightarrow b_1 \rightarrow b_2 \rightarrow ... \rightarrow b_n \rightarrow e_{top} \). Finally, the probability of each consequence can be calculated with:

\[
R(c_i) = R(e_{top})\sum_{s \in S} [P(s)[\prod_{i=1}^{n} P(b_i|s)]]
\]

where \( R(c_i) \) is the frequency of consequence \( c_i \in N_c \) after passing through any recovery barriers \( b_i \in N_b \) such that \( e_{top} \rightarrow b_1 \rightarrow b_2 \rightarrow ... \rightarrow b_n \rightarrow c_i \).

If the state is not known uniquely, then each potential state \( s \in S \) must be enumerated and a probability value \( P(s) \) must be assigned such that \( \sum_{s \in S} P(s) = 1 \). At design-time, this set of states must describe the entirety of the expected operating environment. When the system is deployed at runtime, observations about the current state can be used to refine the set of prior probabilities \( P(s) \) and dynamically calculate current risk values. The equations outlined here use a discrete treatment of probability, but continuous distributions can be used as well where summation operations are replaced by an appropriate integration. If the state can be identified uniquely to a particular state \( s_0 \in S \), then we can assign \( P(s_0) = 1 \) and simplify the risk equations. For example, we could calculate the rate of occurrence for events \( TOP \) and \( C1 \) which are part of the BTD \( B \) shown in Fig. 2 using the following equations:

\[
R(TOP|s_0) = B.f_e(T1,s_0) \ast (1 - B.f_b(B1,s_0)) + B.f_e(T2,s_0) \ast (1 - B.f_b(B2,s_0))
\]

\[
R(C1|s_0) = R(TOP|s_0) \ast [1 - B.f_b(B3,s_0)]
\]

D. Estimating Conditional Relationships

1) Conditional Relationships: In ReSonAte, both the rate of occurrence of events and the effectiveness of barriers may be conditionally dependent on the state including system failure modes, runtime monitor values, and environmental conditions. For each of these categories, it is necessary to identify the factors which should be examined for their impact on this conditional relationship. A failure analysis should be performed using an appropriate failure modeling language as discussed in Section [III-A]. Similarly, the environmental parameters relevant to the system must be identified and the operating environment defined in terms of bounds and expected distributions of these parameters. We assume the environmental conditions are known uniquely and provided to ReSonAte. Monitor values that may impact the conditional relationships should also be identified.

For some nodes in a BTD it may be possible to analytically derive the appropriate conditional relationship with each state variable, but often this relationship must be inferred from data. In this section, we consider a generic threat to the top event chain with a single barrier described as \( t \rightarrow b \rightarrow e_{top} \). The contribution to the rate of the top event \( e_{top} \) from this singular threat \( t \) with a single barrier \( b \) is shown in Eq. (3). Normally, multiple threats can lead to the top event and the contribution of all threats must be considered as described in Eq. (2). However, if all other threat conditions can be eliminated, then the top event may only occur as a result of threat \( t \). In this case, the probability of success for barrier \( b \) can be calculated as a function of the ratio of frequencies between the top event and threat \( t \) shown in Eq. (4). This approach can also be used for threats with multiple associated barriers by considering each barrier in isolation since individual barriers are assumed to be independent in Eq. (1).
\[ R(\epsilon_{\text{top}}|s) = R(t|s) \times (1 - P(b|s)) \]  (5)
\[ R(t_j|s) = 0 \quad \forall \ t_j \in N_t \mid t_j \neq t \rightarrow R(\epsilon_{\text{top}}|s) = R(\epsilon_{\text{top}}|s) \]  (6)
\[ P(b|s) = 1 - [R(\epsilon_{\text{top}}|s)/R(t|s)] \]

We isolate individual threats in simulation using a custom Scenario Description Language (SDL), described in Section III-D2 to generate scenarios where only the one threat of interest is allowed to occur and all other possible threats are eliminated. Each time the threat \( t \) is encountered, the top event \( \epsilon_{\text{top}} \) may either occur or not occur. If the top event does not occur, then the associated barrier \( b \) was successful - i.e., prevented hazard propagation along this path. Otherwise, the barrier was unsuccessful. Since the occurrence of a threat is a discrete event that results in a boolean outcome (i.e. top event does/does not occur), the barrier effectiveness can be modeled as a conditional Binomial distribution where the probability of barrier success is dependent on the ratio of top event frequency to threat frequency as shown in Eq. (6).

For each barrier \( b \) in the BTD, any state variables which are likely to impact the conditional frequency or probability of that node should be identified, and we denote this reduced set of state variables as \( S^b \). For discrete state variables (e.g., presence or absence of a particular fault condition, urban/rural/suburban environment, etc.), this ratio can be estimated using Laplace’s rule of succession [39] as shown in Eq. (7) where \( n_{s_i=a} \) is the number of scenes where the state variable \( s_i \) had the desired value \( a \) and \( k_{s_i=a} \) is the number of such scenes where the top event also occurred. This equation can be applied for each of the possible values of the state variable \( s_i \) to estimate the discrete probability distribution \( P(b|s_i) \), and the process can then be repeated for each relevant state variable \( s_i \in S^c \). Eq. (6) can be used to fuse the estimated distributions of each individual state variable into one multivariate distribution \( P(b|s) \). Similar to Naïve Bayes classifiers, this equation assumes each of the state variables \( s_i \) are mutually independent conditional on the success of barrier \( b \). If a stronger assumption is used that the state variables in \( S^b \) are mutually independent, then the term \( \Pi_{j=0}^{m} [P(s_j)] P(s_0s_1...s_m)^{-1} \) reduces to 1 as was the case for all of the probabilities estimated in our examples.

\[ P(b|s_i = a) = 1 - \frac{k_{s_i=a} + 1}{n_{s_i=a} + 2} \]  (7)
\[ P(b|s) = \frac{P(s_0, s_1, ..., s_m|b)P(b)}{P(s_0, s_1, ..., s_m)} \]
\[ P(b|s_0, s_1, ..., s_m) = \frac{P(b)\Pi_{j=0}^{m}P(s_j|b)}{P(s_0, s_1, ..., s_m)} \]
\[ P(b|s_0, s_1, ..., s_m) = \frac{P(b)\Pi_{j=0}^{m}P(s_j|s_i)P(s_i)}{P(b)^{m-1}P(s_0s_1...s_m)} \]  (8)

For each continuous state variable \( s_j \) (e.g. output of an assurance monitor), maximum likelihood estimation was used in place of Eq. (7) to estimate \( P(b|s_j) \). Similarly, Eq. (8) can be revised for continuous values by replacing the probability mass functions \( P(s_j) \) with probability density functions \( p(s_j) \).

![Fig. 3: This listing shows a fragment of a CARLA scene description that was generated using our SDL written in textX meta language.](image)

While the conditional probability estimation process is outlined here for prevention barriers, it may be modified for recovery barriers by replacing occurrences of any threats \( t \) with the top event \( \epsilon_{\text{top}} \) and replacing occurrences of the top event with each consequence of interest.

2) Scenario Description Language: Description and generation of scenarios that cover the full range of expected operating environments is an important aspect for the design of CPS. This is particularly true for the conditional probability estimation process described above which relies on observing the effects of the various state variables through simulated data. Several domain-specific SDLs such as Scenic [40] and MSDL [41] with probabilistic scene generation capabilities are available. While these languages have powerful scene generation capabilities, they are targeted specifically at the automotive domain. We have developed a simplified SDL using the textX [42] meta language to generate varied scenes for multiple domains including our AV and UUV systems.

A fragment of the scene description for the CARLA AV example is shown in Fig. 3. A scene \( S = \{e_1,e_2,...,e_i\} \) is described as a collection of entities (or set points), with each entity representing information either about the ego vehicle (e.g., type, route, etc.), or the operating environment (e.g., weather, obstacle). Further, each of these entities has parameters whose value can be sampled using techniques such as Markov chain Monte Carlo to generate different scenes in the simulation space. The larger the number of sampling, the wider is the simulation space coverage. For the AV example, our scene was defined as \( S = \{\text{town_description}, \text{weather_description}, \text{av_route}\} \). While the parameters of town_description and av_route remained fixed, the parameters of weather_description such as cloud, precipitation, and precipitation deposits were randomly sampled to take a value in \([0,100]\). We generated 46 different CARLA scenes by randomly sampling the weather parameters, a few of which were used for estimating the conditional relationships.

E. Dynamic Assurance Case Evaluation

The estimated hazard rates can be supplied as evidence to complete the hazard analysis branch of the assurance case shown in Fig. 4. In this example, the top level goal, G1, that potential safety risks have been identified and sufficiently mitigated. This goal is divided into two sub goals: (G2) that all foreseeable hazards have been identified and (G3) the
identified hazards have either been eliminated or sufficiently mitigated. For this example, we restrict our analysis to a single hazard condition regarding a collision due to a roadway obstruction as noted in goal G4. Note that a more comprehensive risk reduction argument would likely consider many other hazard conditions which are represented in this example by the uninstantiated goal G5. Results from the hazard identification process are used as evidence in solution S1. The BTD based hazard rate is supplied as the evidence for the solution S2. Further, goal G5 has also been marked as undeveloped since methods other than BTDs may be used to provide evidence.

Further, if the static hazard rates are available, the hazard analysis branch of the assurance case can be evaluated to ensure the dynamic hazard rates of all the identified hazards are below acceptable static rate thresholds, which are computed during design-time. If the dynamically calculated rates remain below this threshold for all hazards, then the assurance case remains valid and the system is sufficiently safe. Otherwise, the assurance case is invalid and a contingency plan should be enacted to put the system into a safe state.

IV. Evaluation

We evaluate ReSonAte using an AV example in the CARLA simulator [14] and show its generalizability with preliminary results from an UUV example [43]. The experiments in this section were performed on a desktop with AMD Ryzen Threadripper 16-Core Processor, 4 NVIDIA Titan Xp GPU’s and 128 GiB memory.

A. Ground Autonomous Vehicle

1) System Overview: Our first example system is an autonomous car which must safely navigate through an urban environment while avoiding collisions with pedestrians and other vehicles in a variety of environmental and component failure conditions. The architecture of our AV, shown in Fig. 6, relies on a total of 9 sensors including two forward-looking radars, three forward-looking cameras, a Global Positioning System (GPS) receiver, an Inertial Measurement Unit (IMU), and a speedometer. The "Navigation" LEC, adapted from previous work [44], produces waypoints for the desired position and velocity of the vehicle at a sub-meter granularity using a neural network for image processing along with higher-level information about the desired route provided by a map. These waypoints are passed to the "Motion Estimator" which computes throttle and steering angle error between the current and desired waypoints. The Motion Estimator also serves as a supervisory controller which will override the primary Navigation component if an alarm is sent by the AEBs safeguard component. This AEBs component will raise an alarm when the vehicle safe stopping distance, estimated based on the current vehicle speed, exceeds the distance returned by the radars indicating that the AV is approaching an object at an unsafe speed. Finally, the output of the Motion Estimator is sent to two PID controllers which generate appropriate steering, throttle, and brake control signals.

2) System Analysis: We start with the analysis of the AV system and its operating environment (e.g., weather, traffic,
etc). As the AV primarily uses a perception LEC, parameters such as weather conditions (e.g., cloud spread, precipitation level, and precipitation deposit level), high brightness, and camera related faults such as blur and occlusion influenced the control actions generated by the LEC controller. To detect the camera faults and adverse operating conditions, we have designed several monitors. Each of the three cameras is equipped with OpenCV based blur and occlusion detectors to detect image distortions. The blur detector uses the variance of the laplacian [45] to quantify the level of blur in the image where a high variance indicates that the image is not blurred, while a low value (<30) indicates the image is blurred. The occlusion detector is designed to detect continuous black pixels in the images, with the hypothesis that an occluded image will have a higher percentage of connected black pixels. In this work, a large value (>15%) of connected black pixels indicated an occlusion. Additionally, the LEC controller has a reconstruction based $\beta$-VAE assurance monitor [11] for identifying other changes in the operating scenes (e.g., high brightness). A $\beta$-VAE with hyperparameters $\beta=1.2$ and 100 latent units is trained on 6000 images from CARLA scenes of both clear and rainy scenarios. The reconstruction mean square error of the $\beta$-VAE is used with Inductive Conformal Prediction [46] and power martingale [47] to compute a martingale value. TFPG models for the identified camera faults were constructed, but the TFPG reasoning engine was not used since the available monitors were sufficient to uniquely isolate fault conditions without any additional diagnostic procedures.

3) Hazard Analysis and BTD Modeling: For our AV example, we consider a single hazard of a potential collision with roadway obstructions. The potential threats for this hazard were identified to be pedestrians crossing the road (T1), and other vehicles in the AV’s path of travel (T2). The top event was defined as a condition where the AV is approaching a roadway obstruction with an unsafe speed. The primary LEC is trained to safely navigate in the presence of either of these threats and served as the first hazard control strategy for both threat conditions, denoted by prevention barriers B1 and B2. Finally, the AEBS served as a secondary hazard control strategy denoted by the recovery barrier B3. The BTD shown in Fig. [2] was constructed based on these identified events and control strategies. For full-scale systems, the BTD construction process will usually result in the identification of a large number of events and barriers modeled across many BTDs. For our example, we have restricted our analysis to these few events and barriers contained in a single BTD.

4) Conditional Relationships: Here we consider the barrier node B2 in Fig. [2] as an example for the probability calculation technique described in Section III-D. Barrier B2 is part of the event chain $T2 \rightarrow B2 \rightarrow TOP$ and describes the primary control system’s ability to recognize when it is approaching a slow-moving or stopped vehicle in our lane of travel (T2) too quickly and slow down before control of the roadway obstruction hazard (H1) is lost. To isolate barrier B2, simulation scenes were generated using our SDL where T2 was the only threat condition present but all other system and environmental parameters were free to vary. It is important for these scenes to cover the range of expected system states and environmental conditions since the resulting dataset is used to estimate the conditional probability of success for barrier B2. A total of 300 simulation scenarios were used for estimating the probability of barrier B2. As a convenience for our example, the threat (T2) was set to occur once during each scene, regardless of other state parameters, which allows the denominator of this ratio to be set as $R(T2|s) = 1$ occurrence per scene.

As perception LEC is the primary controller, the success rate of barriers B1 and B2 will be dependent on the image quality. The image quality will in turn be dependent on image blurriness, occlusion, and environmental conditions. While the level of blur and occlusion for each camera is a configurable simulation parameter, our example system is not provided this information and instead relies on blur and occlusion detectors for each of the three cameras as discussed in Section IV.A.2 Each detector provides a boolean output indicating if the level of blur or occlusion exceeds a fixed threshold. The primary LEC is also susceptible to OOD data, and an assurance monitor was trained for this LEC to detect such conditions.

This results in barrier B2 being dependent on a total of 7 state-variables described as the set $S^{B2}$. For the 6 boolean variables, Laplace’s rule of succession shown in Eq. [7] was applied to the simulation dataset resulting in probability table in the lower-right section of Fig. [2]. A sigmoid function was chosen to model the conditional relationship with the continuous assurance monitor output. Maximum likelihood estimation was used to produce the function $P(x|s.m.LEC)$ shown in Fig. [2] where $s.m.LEC$ represents the output of the LEC assurance monitor. Each of these single variable functions was combined into the multivariate conditional probability distribution $f_b(B2, s)$ using Eq. [8]. Note that the LEC was observed to be similarly effective in identifying pedestrians and vehicles, and a simplifying assumption was made to use the same conditional probability function for both barriers B1 and B2 given by function $f_b(x = (B1,B2), s)$ in Fig. [2].

The AEBS described by barrier B3 is independent of the camera images and relies on the forward looking radar. The level of noise in our simulated radar sensor increased with increasing precipitation levels, indicating that the effectiveness of barrier B3 would likely decrease as precipitation increased. Also, failure of the radar sensor may occur on a random basis which reduces the effectiveness of the AEBS to zero. A similar conditional estimation process as used for B2 was applied here to calculate the function $f_b(B3, s)$ shown in Fig. [2].

5) Results: To validate the ReSonAte framework, the AV was tasked to navigate 46 different validation scenes that were generated using our SDL. In these scenes, the weather_description parameters of cloud (c), precipitation (p), and precipitation deposits (d) were varied in the range [0,100]. Other adversities were synthetically introduced using OpenCV including increased image brightness, camera occlusion (15%-30% black pixels), and camera blur (using 10x10 Gaussian filters). During each simulation, the ReSonAte’s risk calculations continuously estimate the hazard (or collision) rate $h(t)$ based
on changing environmental conditions, presence of faults, and outputs from the runtime monitors. The frequency of the risk estimation can be selected either based on the vehicle’s speed, environmental changes, or available compute resources. In our experiments, we estimate the risk every inference cycle.

Fig. 7 shows the estimated collision rate and the likelihood of collision as the AV navigates 2 validation scenes. The occurrence of a collision can be described as a random variable following a Poisson distribution where the estimated collision rate is the expected value \( \lambda \). The likelihood of collision can then be computed as \( (1 - e^{-\lambda t}) \), where \( \lambda \) is the estimated collision rate and \( t \) is the operation time which is fixed to 1 minute in our experiments. The scene description and the fault modes are provided in the figure caption.

Fig. 8 shows the estimated average collision rate plotted against the observed collisions for 6 validation scenes as described in the figure caption. The estimated average collision rate is calculated as \( \frac{\int_{T_1}^{T_2} h(t) dt}{T_2 - T_1} \), where \( h(t) \) is the estimated collision rate, \( T_1 = 0 \) and \( T_2 = 1 \) minute for our simulations. An overall trend in the plot shows a strong correlation between the actual and the estimated collisions. Also, a visible trend is that the collision rate changes with the weather patterns, increasing for adverse weather conditions (S5-S6). However, the estimated risk tends to slightly overestimate when the collision rate is greater than 1.0.

Further, each of the 608 data points shown in Fig. 9a represents the outcome of one simulated scene with the actual number of collisions plotted against the average collision rate estimated by ReSonAte. Since the occurrence of a collision is a probabilistic event, there is significant variation in the actual number of collisions observed in each scene. Using our dynamic rate calculation approach, the rate parameter \( \lambda \) of the Poisson distribution changes for each scene as shown on the x-axis of Fig. 8. Maximum likelihood analysis was used to compare our dynamic approach against a static, design-time collision rate estimate where \( \lambda \) is fixed for all scenes. The observed average collision rate across all scenes was found to be 0.829 collisions per minute. Using this static \( \lambda \) value gave a log likelihood of -740.7 while the dynamically updated \( \lambda \) resulted in a log likelihood of -709.8 for a likelihood ratio of 30.9 in favor of our dynamic approach. Note that the static collision rate used here is a posterior estimate calculated from the true number of collisions observed. Any static risk estimate made without this post facto knowledge would result in a lower likelihood value and further increase the gap between the dynamic and static approaches.

To better show the correlation between estimated and actual collisions, the same data points have been divided into bins and averaged in Fig. 9b along with two least-squares fit trend lines. The dashed green trend line shows a linear fit to the complete data set while the dotted red line shows a linear fit to only those data points where the average estimated collision rate was less than or equal to 1.0. Both trend lines show a strong positive correlation between the estimated collision rate and the number of observed collisions, but the dotted red line more closely resembles the desired 1-to-1 correspondence between estimated and actual collisions (i.e. slope of 1). These results indicate that our dynamic risk calculation tends to overestimate when the estimated collision rate is greater than 1.0.
Table I shows the resource requirements and execution times for system with different configurations. As seen from the shaded columns, the additional sensors, and system monitors, particularly the resource intensive β-VAE monitor, increases the GPU memory used by ∼40% and execution time by ∼0.2 seconds. However, the ReSonAte risk calculations require minimal computational resources taking only 0.3 milliseconds.

B. Unmanned Underwater Vehicle

In this section we discuss the primary results of applying ReSonAte to a UUV testbed based on the BlueROV2 [43] vehicle. The testbed is built on the ROS middleware [48] and simulated using the Gazebo simulator environment [49] with the UUV Simulator [50] extensions.

1) System Overview and BTD Modeling: In this example, the UUV was tasked to track a pipeline while avoiding static obstacles (e.g., plants, rocks, etc.) as shown in Fig. 5b. The UUV is equipped with 6 thrusters, a forward looking sonar (FLS), 2 side looking sonars (SLS), an IMU, a GPS, an altimeter, an odometer, and a pressure sensor. The vehicle has several ROS nodes for performing pipe tracking, obstacle avoidance, degradation detection, contingency planning, and control. Additional contingency management features such as thruster reallocation, return-to-home, and resurfacing are available. The UUV uses the SLS along with the FLS, odometry, and altimeter to generate the HSD commands for pipe tracking and obstacle avoidance. The degradation detector is an LEC which employs a feed-forward neural network to detect possible thruster degradation based on thruster efficiency information. This LEC sends information to the contingency manager including the identifier of the degraded thruster, level of degradation, and a value measuring confidence in the predictions. The contingency manager may then perform a thruster reallocation (i.e. adjustment of the vehicle control law) if necessary to adapt to any degradation. A ROS node implementation of ReSonAte was used to dynamically estimate and publish the likelihood of a collision.

2) Hazard Analysis and BTD Modeling: Using the system requirements and its operating conditions we identified UUV operating in presence of static obstacles as the hazard condition which could result in the UUV’s collision. The static obstacle which appears at a distance less than the desired separation distance of 30 meters is considered to be a threat in this example. Further, the static obstacle appearing at a distance less than a minimum separation distance of 5 meters is considered to be a TOP event for the BTD. This information was used to outline a BTD for the UUV example.

3) Conditional Relationships: The probability estimation method described in Section III-D was used to compute the conditional probabilities for the BTD. We used data from 350 simulation scenarios for the conditional probability calculations. These simulations were generated using several scenes generated by varying parameters such as the obstacle sizes (cubes with lengths (0.5,1,2,5,10) meters), the obstacle spawn distance (value in range [5-30] meters), and random thruster failures that were synthetically introduced by varying the thruster1 efficiency in the range [0-60].

4) Results: Fig. 10 shows the ReSonAte estimated likelihood of a collision and the estimated likelihood of collision increases to 0.25. Soon after, the contingency manager performs a thruster reallocation to improve the UUV’s stability, and the likelihood of collision decreases to 0.15. We are currently validating the estimated likelihood across a large number of simulations.

V. CONCLUSION AND FUTURE WORK

ReSonAte captures design-time information about system hazard propagation, potential consequences, and hazard control strategies using BTDs, then uses the information contained in these models to calculate risk at run-time. The frequency of threat events and the effectiveness of hazard control strategies influence the estimated risk and may be conditionally dependent on the state of the system and operating environment. A technique for measuring these conditional relationships in simulation using a custom SDL was demonstrated and a method of evaluating an assurance case using the dynamically estimated risk was described. ReSonAte was applied to an AV example in the CARLA simulator to dynamically assess the risk of collision, and a strong correlation was found between the estimated likelihood of a collision and the observed collisions. Additionally, ReSonAte’s risk calculations require minimal computational resources making it suitable for resource-constrained and real-time CPS applications.

Future extensions and applications for ReSonAte include: (1) dynamic estimation of event severity in addition to event likelihood, (2) inclusion of state uncertainty into risk calculations, (3) forecasting future risk based on expected changes to system or environment, (4) use of estimated risk for higher-level decision making such as controller switching or enactment of contingency plans, and (5) continuous improvement of conditional probability estimates at run-time from operational data.

Acknowledgement: This work was supported by the DARPA Assured Autonomy project and Air Force Research Laboratory.
[48] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in ICRA Workshop on Open Source Software, 2009.

[49] N. P. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems. Citeseer, 2004.

[50] M. M. M. Manhães, S. A. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach, “UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation,” in OCEANS 2016 MTS/IEEE Monterey. IEEE, sep 2016. [Online]. Available: https://doi.org/10.1109%2Foceans.2016.7761080