Model-driven Development of Safety Architectures

Ewen Denney, Ganesh Pai, and Iain Whiteside
SGT / NASA Ames Research Center
Moffett Field, CA 94035, USA
{ewen.denney, ganesh.pai, iain.whiteside}@nasa.gov

***Abstract*—We describe the use of model-driven development for safety assurance of a pioneering NASA flight operation involving a fleet of small unmanned aircraft systems (sUAS) flying beyond visual line of sight. The central idea is to develop a *safety architecture* that provides the basis for risk assessment and visualization within a *safety case*, the formal justification of acceptable safety required by the aviation regulatory authority. A safety architecture is composed from a collection of *bow tie diagrams* (BTDs), a practical approach to manage safety risk by linking the identified hazards to the appropriate mitigation measures. The safety justification for a given unmanned aircraft system (UAS) operation can have many related BTDs. In practice, however, each BTD is independently developed, which poses challenges with respect to incremental development, maintaining consistency across different safety artifacts when changes occur, and in extracting and presenting stakeholder specific information relevant for decision making. We show how a safety architecture reconciles the various BTDs of a system, and, collectively, provide an overarching picture of system safety, by considering them as views of a unified model. We also show how it enables model-driven development of BTDs, replete with validations, transformations, and a range of views. Our approach, which we have implemented in our toolset, AdvoCATE, is illustrated with a running example drawn from a real UAS safety case. The models and some of the innovations described here were instrumental in successfully obtaining regulatory flight approval.**

***Index Terms*—Bow tie diagram, Model-driven development, Safety architecture, Safety case, Transformation, Unmanned aircraft systems, Views**

I. Introduction

Currently small unmanned aircraft systems (sUAS) (popularly referred to as *drones*) are primarily only permitted to fly within visual line of sight of their pilots and under specific operational restrictions, e.g., over unpopulated areas, and at low altitudes, to ensure safety of third parties in the air and on the ground [1].

A capability that will substantially expand the kinds of operations that can be conducted with sUAS—e.g., package delivery, reconnaissance and surveillance, inspection of tall and/or remote structures such as pipelines, power lines, etc.—involves so-called *beyond visual line of sight* (BVLOS) flight, where unmanned aircraft (UAs) fly substantially farther than the visual range of their pilots, and/or their visual observers. These are an entirely new class of operations that are currently the focus of intense industrial interest. However, they pose greater safety risk, especially when conducted in denser airspace in the presence of non-participating air traffic, and over populated areas. Conducting BVLOS operations currently requires approval from the Federal Aviation Administration (FAA), the US national civil aviation authority, which is granted on sufficiently justifying the safety of the proposed operations through a *safety case*.

In short, this is a type of safety risk management artifact addressing, at a minimum, *a*) the details about the system and its environment, including pre-existing procedures, operations, roles, and responsibilities; *b*) the intended changes to the system, e.g., the introduction of new technology, equipment, and procedures; *c*) information on UAS capabilities and airworthiness[1]; *d*) hazard and risk analyses, including details of the assumptions made, the criteria for categorizing hazards, the initial and residual risk, hazard mitigations, risk treatment and safety requirements, and hazard tracking; and *e*) details of safety risk management planning.

An important component of the safety case is a justification for how the specified hazard mitigation measures and safety requirements are expected to reduce risk to an acceptable level. A practical approach to present some key parts of this safety justification employs bow tie diagrams (BTDs), which elaborate the safety-relevant scenarios and the suite of measures used to manage safety risk. BTDs have been used for safety risk management in the context of both civil aviation [2] as well as UAS [3]. Creating an aviation safety case (and especially the required safety justification) involves substantial engineering effort [4], [6], [7], [8], and we believe model-driven development can play a useful role.

Recently, we created a safety case to enable BVLOS flight trials with a fleet of sUAS, as part of the effort to engineer an air-traffic management system specifically for low-altitude sUAS operations [9]. The safety case harnessed a diversity of engineering artifacts, analyses, and evidence, along with BTDs to provide the justification of risk reduction and, in turn, operational safety. Using the BTDs and some of the innovations to be described subsequently in this paper, we conveyed, to the FAA, the overall picture of safety risk together with the measures that would be undertaken to ensure the intended operations would be safely conducted. The safety case also employed structured argumentation to provide assurance of functional safety for the required surveillance and avoidance capabilities [4], [6]. This safety case successfully underwent regulatory scrutiny, and served as the basis for flight approval. So far as we are aware, the safety case and the associated operations are the first of their kind—i.e., true BVLOS operations in non-segregated airspace over (sparsely)

[1]Fitness of an aircraft system to safely initiate, sustain, and terminate flight.

populated land within the US, with multiple sUAS—to have been granted regulatory approval.

As part of this effort, we encountered a number of practical challenges with creating and using BTDs, as they may be typically used. In brief, those challenges pertain to incremental development, maintaining consistency across both different BTDs and other safety artifacts when changes occur, determining that the *safety system* represented by the BTDs adequately reduces risk, and extracting and presenting stakeholder-specific information relevant for safety-related decision making.

This paper describes our novel solutions to address these challenges, and our contributions include:

a) a refinement of *safety architectures* (SA)—introduced in our earlier work [4] and subsequently formalized [5]—that reconciles a collection of BTDs within one instance of a unifying model where BTDs emerge naturally as substructures;

b) the specification of a series of validations, views, and transformations that enable model-driven development (MDD) of SA. We used the SA and some of the views to extract and provide specific assurance information, e.g., an assessment of residual risk, which was required for regulatory scrutiny;

c) enhancing the methodology for using BTDs during safety case development. To our knowledge, the views and transformations defined here are new innovations, not otherwise part of how BTDs have been traditionally used. As such, we believe they can also be used in other domains where safety assurance is required; and

d) tool support for the preceding contributions, in our toolset for assurance case automation, AdvoCATE [10].

II. Background

We first give an introduction to BTDs, providing a context for their use during the safety analysis and assurance process. Then we give excerpts of BTDs from the safety case we created as the running example that motivates this work and illustrates our novel solution. Finally, based on this example, we elaborate some of the practical challenges that we encountered in creating and using BTDs to provide safety justification, as part of our wider task of creating a safety case for BVLOS sUAS operations.

A. Safety Analysis and Bow Tie Diagrams

The safety process begins, in general, with hazard identification based upon a particular *concept of operations* (CONOPS), i.e., a description of the system, its boundaries, and a characterization of the intended usage. The resulting *safety hazards* represent the activities, conditions, or the operational contexts that pose the potential for harm.

Then, we establish hazard causes, and (worst-case credible) consequences, after which we identify and elaborate pre-existing mitigations towards determining an initial *risk level*, i.e., a preliminary assessment of the combination of the (expected) likelihood of occurrence of a harmful event, and its associated severity. That, in turn, establishes whether additional mitigation measures are required. If required, and once new mitigations have been determined, a subsequent risk assessment is undertaken to determine a *residual* risk level, i.e., the remaining risk after applying all mitigation measures to reduce inherent risk. This overall process is iterative until a residual risk level has been achieved that is considered acceptable (and, therefore, safe) [11]. These activities also form part of the core process for developing a safety case in the context of sUAS [4].

There are mappings from the outcomes of the safety process into the elements of a BTD. As shown in Fig. 1, a BTD provides a means to visualize the links between the identified hazards, and their causes, consequences, and the required mitigation measures. From the BTD perspective, a *hazard* captures an operational context, while the *top event* represents the state where there is a loss of control over the hazard, or a hazard release. In Fig. 1, the hazard identified from our safety case is H1: *Airborne UAs operating BVLOS operations within the operating range* (OR). Here, the OR is a defined volume of airspace within which BVLOS sUAS operations will occur. This is an inherently hazardous activity since the possibility of colliding with other (manned or unmanned) air traffic poses the potential for harm. A corresponding top event is a loss of safe separation between the UA and the manned aircraft (e.g., E4, Fig. 1). A *consequence* (e.g., E7, Fig. 1) is a harmful event such as a midair collision (MAC), whereas a *threat* represents a potential cause of a top event, e.g., an *airborne intruder* (E3, Fig. 1).

Barriers represent (pre-existing or new) risk mitigation measures. We can further refine barriers into their constituent *controls*[2], i.e., lower-level actions/capabilities that collectively implement the safety function that the barrier provides. From Fig. 1, it is evident that we organize barriers such that their combination works to *i*) prevent the progression of the event chains leading from hazard causes to the events where there is a dangerous loss of control over the system, i.e., hazard release; and *ii*) recover from hazard release to prevent further progression of the event chain from reaching harmful consequences. Thus, barriers appearing to the left of the top event are meant for *prevention*, i.e., to reduce the *likelihood* of occurrence of the top event. Alternatively, the purpose of the barriers to the right of the top event is *recovery*, i.e., to reduce the *risk* of the consequence *after* the top event has occurred.

The BTD of Fig. 1 gives specific prevention and recovery barriers[3] that were pertinent to our safety case, e.g., an *independent flight abort* capability, and *piloting safety actions*, respectively. Also, in Fig. 1 we elaborate the specific, relevant control(s) below a given barrier. For instance, notifying air traffic control (ATC) and broadcasting on the common traffic advisory frequency (CTAF) are amongst the many controls comprising the *emergency procedures* (prevention) barrier. Notice that we can use the same barrier for both prevention

[2]The terms *barrier* and *control* are often used interchangeably in the literature, although we distinguish them here.

[3]Although representative of the real suite of safety measures being used, the BTDs in this paper are mainly illustrative and, therefore, do not show all the barriers.

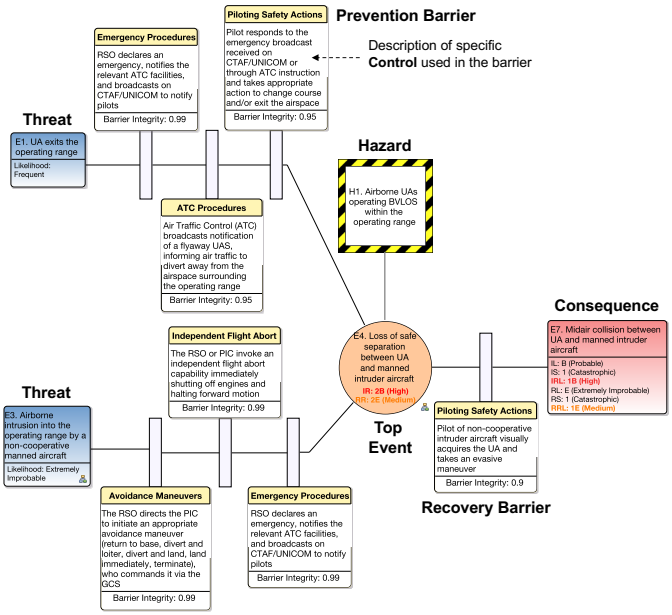


Fig. 1. Excerpt of a BTD from a real safety case illustrating its structure and key elements (annotated in **boldface**), as implemented in our tool, AdvoCATE. This BTD concerns the mitigation of a *loss of separation* event that could lead to a *midair collision* consequence, in the context of specific threats, and UAs flying BVLOS within a defined operating airspace.

and recovery, but that their constituent controls are different, e.g., as in the *piloting safety actions* barrier in Fig. 1. Barriers/controls can be further developed at a second level (not shown in Fig. 1) into *escalation factors* and *escalation factor barriers*, though we will not consider them in this paper.

In short, BTDs present a visualization of the key concerns relevant for safety, along with the measures available to reduce the associated risk. Thus, it is intuitive to see how we can use BTDs for safety justification within an aviation safety case. Implicitly, that safety justification can be considered as providing *defense in depth* against the identified causes of specific harmful consequences through the use of independent, loosely-coupled *barriers* or, equivalently, *layers of protection*.

Next, we introduce two additional BTDs from the safety case we authored. Together with the BTD of Fig. 1, the BTDs of Fig. 2 and Fig. 3 comprise the running example for the rest of this paper.

B. Running Example

As overall context for interpreting this example, we first briefly describe the concept of operations (CONOPS) and the pertinent safety concerns: the flight operations—which are meant to simulate some of the real applications indicated in Section I, whilst testing the capabilities of the UAS traffic management (UTM) system [9]—involve multiple sUAS operating simultaneously within an operating range (OR), over a (sparsely) populated, and minimally built-up area. Some of the UAs operate BVLOS while others operate within line of sight. Two main hazardous situations are relevant:

H1) UAs that are airborne BVLOS within the OR: this scenario is hazardous due to the presence of regular air traffic in the area that operate at a low altitude. That, in turn, poses the potential for a MAC with a manned aircraft intruding into the OR. Note that this is exactly the hazard in Fig. 1, and also in Fig. 2.

H2) So-called non-cooperative, non-participating aircraft[4] that are airborne in a *traffic pattern* outside the OR, but within the *threat volume* (TV)[5]: this hazard is captured in the BTD of Fig. 3, and the scenario is a specialization of the previous scenario. It deserves elaboration due to the following reasons:

– aircraft in a traffic pattern indicates the presence of aviation activity at low altitudes in proximity to the OR, in particular aircraft that are taking off or landing. Any departure from the traffic pattern in the direction of the OR is hazardous owing to the reduced time to react.
– even if aircraft from the traffic pattern do not intrude into the OR, the possibility of UAs inadvertently exiting from the OR in the direction of this air traffic also poses a catastrophic collision hazard.

The BTDs of Figs. 1 and 2 address *different* top events (E4, and E3, respectively) for the *same* hazard (H1). Specifically, top event E4 of Fig. 1 concerns a *loss of safe separation* with a manned intruder aircraft, whereas top event E3 of Fig. 2 concerns an *airborne intrusion into the OR* by a non-participating manned aircraft. Notice that event E3 in Fig. 2 is identical to the threat event E3 in Fig. 1. Moreover, the consequence event E4 in Fig. 2 (addressing a loss of safe separation) is identical to the top event E4 in Fig. 1. Indeed, the branch connecting the two events have the same barriers/controls in both the BTDs: i.e., *avoidance maneuvers*, *independent flight abort*, and *emergency procedures*. The BTDs in Figs. 1 and 2 also identify additional threats and specific consequence events related to hazard H1.

The BTDs of Fig. 2 and Fig. 3 address the *same* top event (E3) for *different* hazards (i.e., H1 and H2, respectively). Observe that the threat event E2, which pertains to a lack of sUAS crew situational awareness, is a common threat in both BTDs. However, each BTD also contains threats and consequences that are specific to the particular situation that the hazard represents. For instance, E5, in Fig. 2, and E1, in Fig. 3 are dissimilar threats for the same top event. Likewise, the consequence E5 is only relevant in the context of hazard H1 and therefore only appears in the BTD of Fig. 2.

There is also flexibility in how we can deploy barriers and controls. For instance, as mentioned earlier, we can use the same barriers in different BTDs (common event chain). So also, the same barriers can be used for different threats not only in different BTDs but also in the same BTD, e.g., the use of the *ground-based surveillance* barrier, using radar as a control, in Fig. 2 and Fig. 3.

Finally, in the BTDs of Figs. 1–3, we annotate barriers with their *integrity*, i.e., the likelihood that a barrier is compromised

[4]That is, third party, manned aircraft not equipped with operating transponders, due to which they are invisible to ATC radar when operating at a low altitude, below primary radar range.

[5]The TV is a volume of airspace larger than, and including the OR, within which there is a credible likelihood of a MAC.

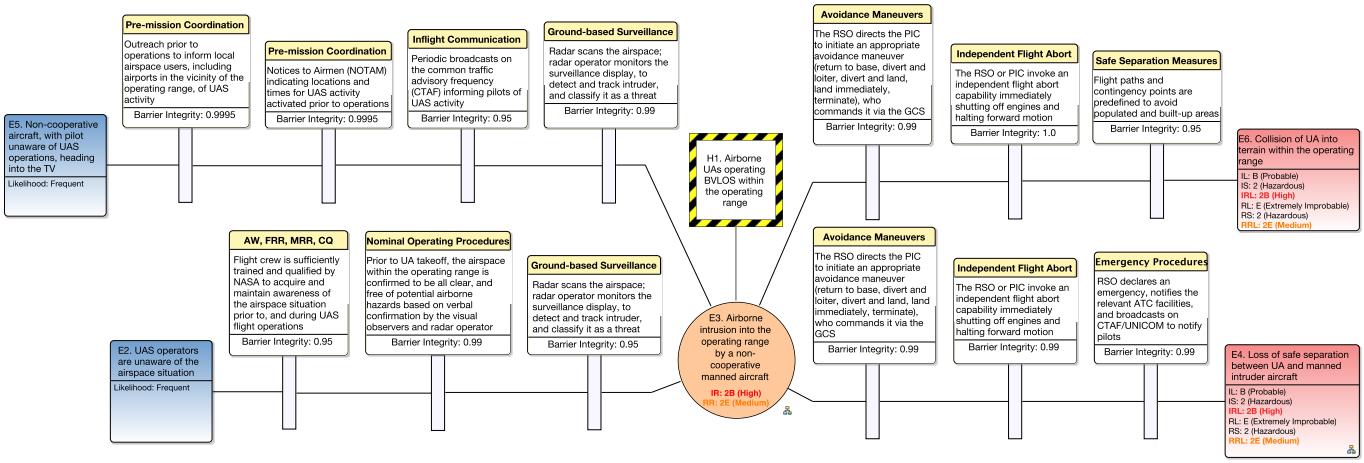


Fig. 2. Excerpt of the BTD to mitigate an airborne intrusion by a manned aircraft into the OR (E3), and eventually prevent either a ground collision of a UA (E6), or a loss of safe separation with the manned intruder (E4).

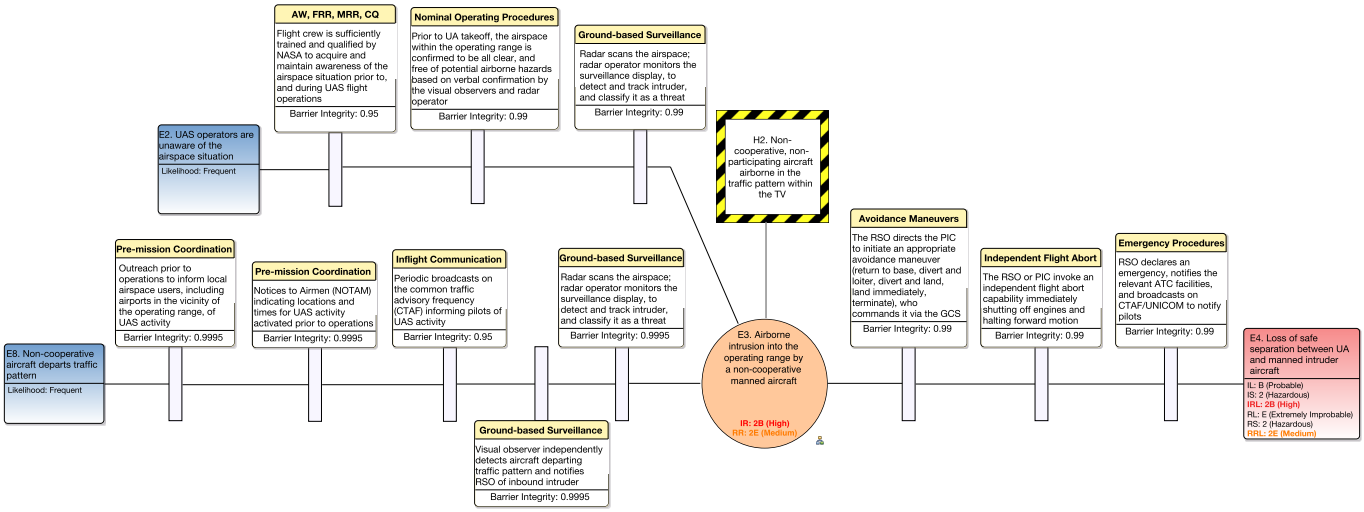


Fig. 3. BTD whose top event (E3) is identical to the BTD of Fig. 2, although in the context of a different hazardous situation (H2) for the same operation, i.e., manned aircraft within a traffic pattern outside the OR that pose a danger if they veer away from the pattern in the direction of sUAS activity.

in a dangerous manner. Barrier integrity is closely related to, but not the same as barrier *reliability*, which concerns all barrier breaches, including those that may not have safety-relevant implications. Based on an initial estimate of threat event likelihood and the severity of the worst-case credible consequence, each BTD provides an assessment of initial and residual risk. In general, initial risk is the combination of initial severity and likelihood. Residual risk corrects the initial estimate reducing either, or both components of risk based on the barriers used. Then, given the assessment and a traditional risk acceptance and classification matrix [11], we determine *risk levels*. In the BTDs of Figs. 1–3, these are the fields IR and RR, respectively, for the top event (and IRL, and RRL, respectively for the consequence). The RRs of an event are calculated from left to right, based on the RR of its threats and the barriers between them, using the inclusion-exclusion principle to compute the risk of the consequent event.

In reality, a safety case contains several unique hazards and top events each of which have numerous threats, consequences, and several layers of barriers and controls. Intuitively, this presents a number of practical challenges, which we now elaborate.

C. Practical Challenges in Using BTDs

As may have been evident from the preceding discussion, depending on the specific system/operation being considered for safety analysis, we can identify multiple hazardous scenarios, each of which may, in turn, yield multiple top events. This results in several BTDs—one per specified top event per identified hazard. In practice, each BTD is typically individually created, and since there can be inherent relationships between such BTDs—e.g., through common threats, or the use of common barriers/controls for multiple threats—creating each BTD independently presents challenges that pertain to *a*) ensuring that the relevant relations between the different BTDs are captured, and *b*) maintaining internal consistency.

Since the safety process is iterative, each iteration refines our understanding of issues such as appropriately classifying events (e.g., into causes, consequences, top events), event ordering, the relevance and organization of controls/barriers as well as their respective ordering, changes required to existing controls, etc. This necessitates an incremental development of BTDs due to which, again, there is a need to ensure that updates are both internally consistent, and consistent with the underpinning safety analysis.

Moreover, the controls and barriers in a BTD have counterparts in the containing safety case, as well as in the real safety system. For example, nominal and off-nominal procedures and actions to be followed by crew members—which could be specified across multiple BTDs—can be collected and invoked in one location in the safety case, and will additionally exist as actual documents that crew members will reference during the real mission; likewise for hardware/software implementations of the functionality specified by a control/barrier, etc. The challenge of maintaining consistency also extends to these external artifacts to which the BTDs refer. To an extent, this can become as safety-critical a concern as the identified hazards [12].

Whilst creating the safety case for BVLOS sUAS operations, we became keenly aware of the importance of presenting stakeholder-specific information as well as the larger context of safety to facilitate better comprehension and decision making. For example, the regulators were interested in the overarching picture of how the barriers would be used for safety and the extent of risk reduction that would be achieved, sometimes focusing on the details of the controls only for specific barriers, e.g., *radar surveillance*. On the other hand, those stakeholders who are part of the system (as opposed to those evaluating it) were more concerned with their specific responsibilities for ensuring safety in operations. Moreover, particular details relevant for specific stakeholders are, more often than not, spread across a plurality of BTDs, which requires a capability not only to extract this information but also to present it in a form that is relevant for the particular purpose/stakeholder. In relation to these needs, the collection of BTDs can be seen as presenting snapshots of how system safety is being managed, but not the overarching picture.

So far as we are aware, in practice, current methodologies and/or tools available for creating BTDs, e.g., [13], [14], [15], [16], [17], [18], do not offer the capabilities to address the challenges outlined here.

III. Safety Architectures

Now we introduce and motivate our design of safety architectures (SAs). Then, we present the EMF model [19] underpinning our implementation, which, for assurance, is based on our formalization of SAs.

A. Design decisions

We will draw from the running example in Section II-B to illustrate the main design decisions of SAs.

i) We observe from Figs. 2 and 3 that the events, barriers, and controls in one BTD can be used (selectively or entirely) in the other, recalling that the two BTDs address the same top event (E3) for different hazards (H1, and H2, respective), also permitting hazard-specific events and barriers. For example, E8 is a threat only relevant for hazard H2, as are the barriers on that event chain. As such, we consider *events*, *controls*, and *barriers* to have a separate existence outside the BTDs in which they occur. We call this global list a *safety signature*.

ii) We further recall that the BTDs of Figs. 1 and 2 both concern hazard H1, but address different top events (E4, and E3, respectively). We additionally note that the event chain connecting the events E3 and E4 are common to both BTDs, including the barriers/controls used. Intuitively this consistency is required since the events in the context of the same hazard are identical irrespective of where the focus of risk mitigation lies. In other words, the two BTDs can be considered as a moving window on a single event chain and depending on which event is in focus (i.e., considered a top event), other events can be classified appropriately (i.e., as threats, or consequences). In this light, creating BTDs for hazards can be seen as the process of constructing an interconnected graph of events and barriers/controls, drawn from the safety signature. We call this interconnected graph a *Controlled Event Structure* (CES). Fig. 4 shows (as a bird's eye view) the CES that includes the BTDs of Figs. 1 and 2.

iii) Finally, we observe that the risk assessments that accompany the BTDs should be mutually consistent. This necessitates defining an underlying risk assessment model that takes the CES into account rather than individual BTDs. This is particularly important when considering the situation where an event is shared across different hazards, e.g., E3, which is a top event in the BTDs of Fig. 2 and Fig. 3, since a BTD specific risk assessment only provides a part of the assessment of total risk posed. In other words, without considering the overarching event chain, there is a danger of underestimating the level of risk posed.

The combination of safety signature and CES for every hazard is a *safety architecture* (SA). This approach, where (traditional) BTDs emerge as projections from an overarching CES, enables us to automatically pinpoint inconsistencies, view the structure in novel ways (e.g., show which events are affected upon a control breach), and to provide automatic transformations that facilitate incremental BTD development.

Elsewhere [5], we have given a formal definition of SAs, building on our earlier formalization of BTDs [4]. Briefly, we assume a safety signature consisting of underlying sets of events, controls, and barriers. A CES can then be defined as a labeled directed acyclic graph of events and barriers, subject to certain structural conditions. Finally, an SA consists of a set of hazards, and a collection of mutually consistent controlled event structures, one for each hazard. A BTD can then be seen as a specific sub-structure of a CES containing the top event and all connected nodes up to (and including) the first events on either side. We omit the details here and refer the interested reader to [5].

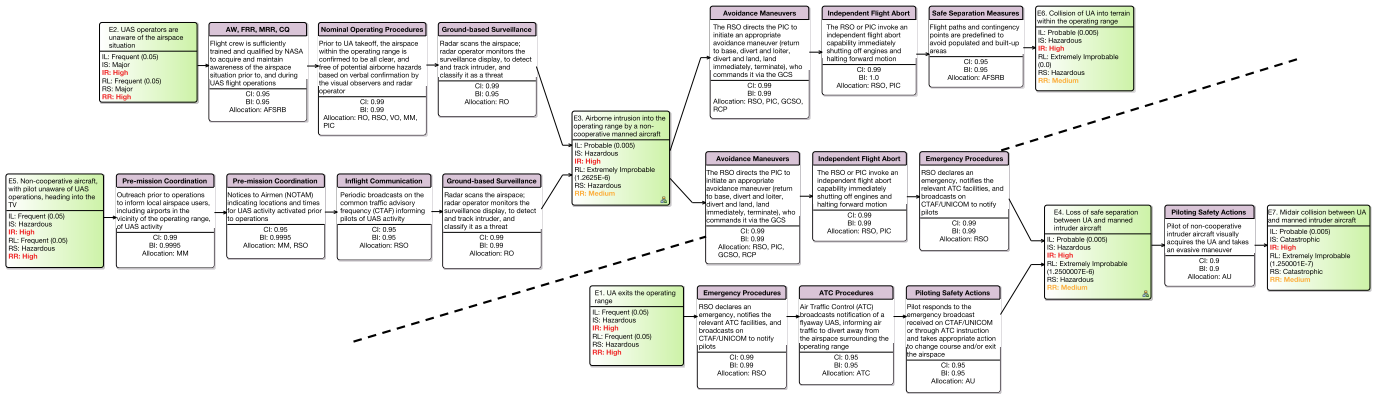


Fig. 4. Example of a CES including two BTDs of the running example. The heavy dashed line splits the CES such that the upper fragment corresponds to the BTD of Fig. 2, and the lower section corresponds to the BTD of Fig. 1. The event chain from event (and including) events E3 to E4 and the barriers between the two are common to both BTDs.

B. Safety Architectures in AdvoCATE

Our tool, AdvoCATE, is an Eclipse RCP application that leverages a variety of models to support the construction of safety cases. All our models are specified using Eclipse Modeling Framework (EMF) [19] and we use Sirius[6] to create our graphical representations. This section presents the EMF model, derived from our formal definition, that underpins our tool and the functionality that it provides

Fig. 5a shows a fragment of the model that represents the safety signature.As mentioned earlier, an SA consists of collections of events, barriers, and controls—the safety signature—and a collection of hazards. These have unique names, along with a description that is presented to the user in the BTDs (e.g., see Fig. 1). Controls belong to a barrier, contain a list of allocations, and have an *integrity* (taking a value between 0 to 1, and which is related to the likelihood of a dangerous breach of a control).

Allocations, such as *Mission Manager*, or *Pilot in Command* represent the person or system responsible for ensuring the integrity of that control. Note that while the theory assures that each control is allocated to a barrier, our model is more 'relaxed', which enables us to model partial SAs. Each hazard is allocated a CES—the graph structure from which our BTDs are derived—and an *associated argument*, which is the main connection between BTDs and the other features of AdvoCATE, detailed in [10].

As shown in Fig. 5b, a CES is a directed acyclic graph (DAG), modeled using a list of nodes and a list of links with *to* and *from* references giving the source and target of each link. We enforce the DAG structure using validations.

Nodes in the CES are *instances* of the events and controls defined in the signature, connected by the *event* and *control* references respectively. Thus, individual events and controls can be reused in different CESs. In fact, as seen, controls can be reused within the same CES.

We allow barrier instances to be defined within a CES, which gives us the flexibility to create logical sub-groupings of controls within the same barrier and CES. We do not distinguish between top, threat, and consequence events, or between prevention and recovery controls in a CES: those properties emerge we *focus* on a particular event as a top event.

Event instances also store *severity* and *likelihood* information for the risk assessment. Our SA model facilitates risk assessment by defining residual severity, likelihood, and risk as *operations*, which derive their values from those of the proceeding elements. The current implementation uses the inclusion-exclusion principle over all of the threat events for a given event. For global threats, i.e., root events in the CES, the calculations for residual levels simply use the initial values.

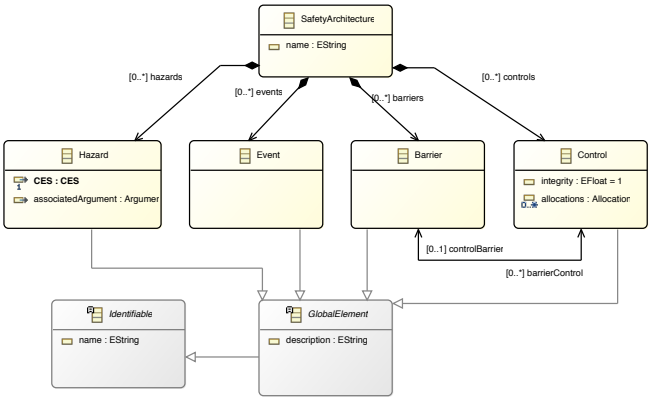
IV. Safety Architecting in AdvoCATE

The progression of model-driven development (MDD) functionality that we will describe in this section can be thought of as an outline of a tool-supported methodology for BTDs, that addresses a number of the practical challenges we identified earlier (Section II-C). First, we discuss how we create safety signatures. Those, in turn, support the creation of BTDs. Then, we define a number of transformations to automate common SA-wide changes, e.g., during development iterations. These are accompanied with validations that ensure consistency. Finally, we specify and illustrate a selection of useful views of a SA, intended to provide stakeholder-specific information.

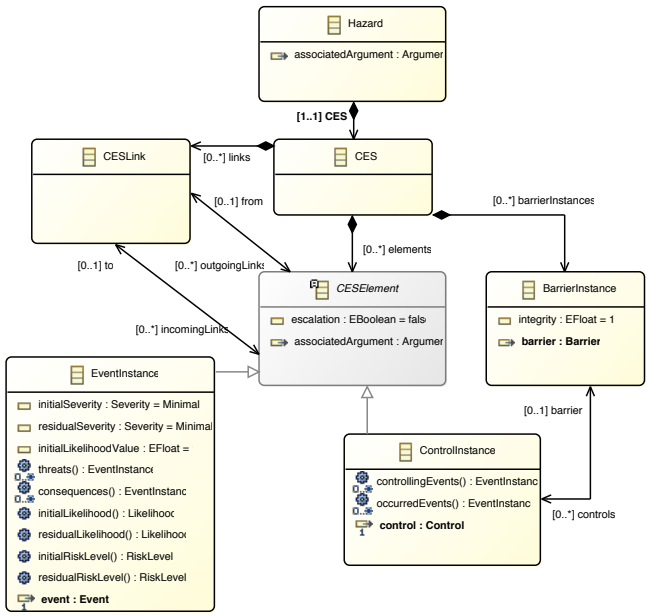
A. Implementation background

Our implementation of BTDs in AdvoCATE can be thought as providing a suite of graphical editors, realized using the Sirius framework, and associated functions that visualize and modify the underlying EMF model. One of the most compelling reasons for using Sirius is the abstraction that it enables between the domain model (i.e., SAs) and any graphical renderings we create. In Sirius, defining a graphical editor can be considered a two-step process: giving mappings from the domain model into a *view specification* and giving a graphical rendering to elements in this view specification. Validations and transformations are defined (respectively) as predicates

[6]URL: https://eclipse.org/sirius/



(a) Safety architecture and safety signature



(b) Controlled event structure

Fig. 5. EMF model for safety architectures.

and functions that operate on an element of the (underlying) model, and seamlessly integrated into the Sirius editors.

In the rest of this section, we formalize an abstract definition of (a subset of) the view mechanism in Sirius and provide an analogous formal definition of *SA transformations*.

1) Views: As described above, a view is defined by a mapping from the domain model to a *view specification*:

Definition 1 (View Specification). A *view specification* is a tuple $\langle \mathcal{N}, \mathcal{L}, s, t, q \rangle$ where $\mathcal{N}$ is a set of node types, $\mathcal{L}$ is a set of link types, and $s, t : \mathcal{L} \to \mathcal{P}(\mathcal{N})$ gives link sources and targets, respectively. The map $q : \mathcal{N} \cup \mathcal{L} \to \mathcal{Q}$ takes node types to *queries*.

This simplified definition omits hierarchical features of views (implemented using containers). Moreover, we abstract away from implementation details, such as data associated with nodes, and the syntax of the query language[7]. For now, we simply assume that we have a family of queries (one for each node and link type, written q_x, where $x \in \mathcal{N} \cup \mathcal{L}$), and that a query is given by a function that takes a node of that type in a given a safety architecture, and returns a set of nodes satisfying the query. The mapping between domain model and view specification is provided by the queries, and execution of a query upon a root object constructs a view.

Definition 2 (View). Given a view specification, $\mathcal{S} = \langle \mathcal{N}, \mathcal{L}, s, t, q \rangle$, and a node r, drawn from the safety architecture, which we call the root element, the '$\mathcal{S}$-*view* of r' is a graph $\langle V, \to, t_n, t_l \rangle$, where $t_n : N \to \mathcal{N}$ gives node types, and $t_l : (\to) \to \mathcal{L}$ gives link types, such that:

a) $v_1 \to v_2$, $t_n(v_1) = N_1$, $t_n(v_2) = N_2$, $t_l((v_1, v_2)) = L$, then $N_1 \in s(L)$ and $N_2 \in t(L)$; that is, the node types match the specification
b) if $v \in V$ and $t_n(V) = N$ then $v \in q_N(r)$. That is, if n has type N then it is in the results of the query for that node type applied to the root element.
c) furthermore, if $v_1 \to v_2$, and $t_l((v_1, v_2)) = L$, then $v_2 \in q_L(v_1)$. Note that the link queries are given in the context of the source, and find a list of targets, each of which has a separate link.

In short, views are given as separately defined graphs (with one for every instance of a root element type), whose nodes and links are defined and populated by the contents of the SA.

2) Validations and Transformations: A validation checks a given property of the model. In Sirius, this is defined as a predicate on a model element. If the predicate is true, a validation marker can be added to the realizations of those elements in any views in which they appear. In AdvoCATE, we also offer transformations that can be applied by the user to make a meaningful domain change to the model. We first give an abstract definition of what we consider a SA transformation, which can be used to prove that application of the transformation on a SA will result in a well-formed transformed SA.

Definition 3 (Safety Architecture Transformation). A *SA transformation*, τ, from a SA (including its signature) $\mathcal{S}_1 = (\Sigma_1, \langle H_1, l_h, ces_1 \rangle)$ to another SA $\mathcal{S}_2 = (\Sigma_2, \langle H_1, l_h, ces_2 \rangle)$ is given by mappings: $\tau_c : C_1 \to \mathcal{P}(C_2)$, $\tau_e : E_1 \to \mathcal{P}(E_2)$, $\tau_b : B_1 \to \mathcal{P}(B_2)$, and $\tau_h : \langle H_1, l_{h_1}, ces_1 \rangle \to \langle H_2, l_{h_2}, ces_2 \rangle$, such that:

- if $c \in b$ and $\tau(c) \in b'$ then $b' \in \tau(b)$
- if $n_1 \to n_1'$ then $\forall n_2 \in \tau(n_1).\ \exists n_2' \in \tau(n_1').\ n_2 \to_2^* n_2'$.

B. Signature Generation

Building BTDs requires connecting events, controls, and barriers from the signature, but creating the signature itself is also supported in AdvoCATE. In fact, we provide two mechanisms for expanding signatures: in situ, while creating BTDs, and separately using a suite of table editors, which enables compact visualization and bulk editing of controls.

[7]Acceleo Query Language: https://eclipse.org/acceleo/documentation/

TABLE I. BTD view specification

	Node Types					
Link Types	*haz*	*threatC*	*conseqC*	*topE*	*threatE*	*conseqE*
hazardL	t			s		
threatL		s, t		t	s	
conseqL			s, t	s		t

TABLE II. Barrier-centric slice view specification

	Nodes		
Links	*BSThreat*	*BSConseq*	*BSControls*
threatL	s		t
conL		t	s

Each table editor is defined by a query that, in turn, defines the rows of the table (e.g., our *controls table editor* uses the *controls* containment reference for the list of controls). The columns of each table are defined by queries on the class represented by each row (*Control* in our case). While these can be arbitrary queries, all of our columns simply project the appropriate fields (and references) in the class: name, description, integrity, and associated barrier. The editor automatically populates references with possible values. Thus, the barrier can be chosen from a dropdown list.

C. BTD Construction

The CES provides a high-level overview of the underlying model, ensuring consistency of each BTD but the primary means of development remains at the level of individual BTDs. AdvoCATE provides a graphical editor for their construction, defined as a view of the CES and whose specification is given according to Definition 1.

For convenience, we present the node and link types for our BTD view specification and their source and target functions in a tabular format (Table I). An s or t in a given cell states that the node type is in the source or target set for that link type. For example, the $threatL$ row states that $s(threatL) = \{threatC, threatE\}$ and $t(threatL) = \{threatC, topE\}$.

Intuitively, the *hazardL* link connects the hazard to the top event, while *threatL* links connect threat events and their controls to the top event,and *conseqL* links connect the top event to the chain of consequences. The queries that populate a view are applied to an *EventInstance* root element. A subset are given, using the query syntax that traverses our model, as given in Fig. 5a and Fig. 5b:

- $q_{haz}(r) = r.eContainer().eContainer()$, where the expression $eContainer()$ is the EMF method for following the class hierarchy and the '.' separator allows chains of calls to reference or fields. That is, given an event instance (r), we navigate to the grandparent class: the desired hazard.
- $q_{topE}(r) = r$, as the top event is the root of the diagram.
- $q_{threatE}(r) = r.threats()$, uses the $threats$ operation (given in the *EventInstance* class in Fig. 5b), recursively following *CESLink*s to find the first preceding events.
- $q_{threatL}(s) = s.outgoingLinks.to$, which gets the to node of all outgoing threats.

Fig. 3 is the BTD view (complying with Definition 2) generated by using event E3 (i.e., *airborne intrusion*) as the root of each query. In addition to the presentation of BTDs, AdvoCATE provides a fully-functional editor to add or remove events and controls. These operations update both the CES and the view.

D. Transformations

The incremental nature of BTD development can require decisions that can affect the overall SA. Similar to programming language refactoring [20], these changes can be both tedious and difficult to achieve correctly, when manually undertaken. AdvoCATE provides transformations that enable such global changes to a SA to be automated. We now describe two transformations that we have found useful.

1) Barrier Splitting and Merging: Occasionally, allocating controls to barriers requires a change in granularity. This can necessitate combining some controls into a single barrier or splitting a combination of controls. For example, in an earlier iteration of the BTD of Fig. 3, the controls related to the *emergency procedures* barrier were split across separate barriers pertaining to ATC notification, and broadcasting on CTAF. As is evident in the figure, they are now part of the same barrier. Formally, we give this as a transformation τ (with parameters, b_1 and b_2, giving barriers to merge) such that τ_e and τ_c are identity transformations, and

$$\tau_b(b) = \begin{cases} b_{new} & b = b_1 \vee b_2 \\ b & otherwise \end{cases}$$

and if $bar(c) \in \{b_1, b_2\}$ then $\tau(bar)(\tau_c(c)) = b_{new}$. Furthermore, the transformation will also modify the CES such that $\tau_{ces}(\langle N, \rightarrow, l, esc\rangle) = \langle N, \rightarrow, l', esc\rangle$ where $l'_t(n) = b_{new}$ whenever $l_t(n) \in \{b_1, b_2\}$, otherwise it is identical.

AdvoCATE provides functionality that can also split a pre-existing barrier into two separate barriers, by giving a partitioning of controls, using a dialog box populated by the choice of barrier to split. Due to space constraints, we omit the formalization of this transformation and present the result of this transformation in Fig. 3. Specifically, the result of a barrier split can be seen in the two prevention barriers identified as *ground-based surveillance* on the event chain between the threat E8 and the top event E3. Although they are labeled with the same description (since, at an abstract level, they both comprise ground-based surveillance), the control in the first barrier concerns surveillance conducted by a visual observer, as distinct and independent from the control in the second barrier, i.e., surveillance conducted using a radar system.

2) Event split: AdvoCATE also offers similar functionality for splitting *events*. Here, we allocate the threat and consequence paths to one, the other, or both of the resultant events. There are two possibilities for event splitting in AdvoCATE: *a*) *sequential* splitting, where the newly split event follows the original event in the event chain; and *b*) *parallel* splitting, where the two events are not connected. Note that in parallel splitting, we can also connect an event to a different hazard than the one from which it was originally split. Conceptually, this corresponds to a different form of increasing granularity of an SA. Again, due to space constraints, we omit the

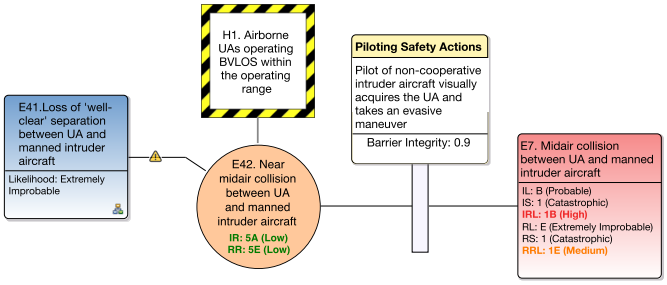


Fig. 6. One of the two BTDs resulting from the *sequential split transformation* applied to the top event E4 of the BTD in Fig. 1. A validation warning informs the user that there are missing controls between the events newly created after the transformation.

formalization of event splitting and present a domain-specific discussion of the results.

For instance, we can split the event E4 in Fig. 1, concerning a *loss of safe separation* into two sequential events, namely, a *loss of well-clear separation* (E41), and *a near midair collision* (NMAC) (E42). Fig. 6 represents one of the BTDs resulting from the transformation, where the original consequence (E7) is retained, but is now a consequence of the new event E42. In the other resulting BTD (not shown), the event E41 is now the top event, and retains the original threats E1 and E3 of Fig. 1. The reason for this kind of transformation here corresponds to two different time thresholds where we can provide finer grained reaction to recover before a MAC occurs.

Indeed, if an automated collision avoidance system were available onboard the UA, it would rely on an alert of a loss of well-clear separation before engaging a collision avoidance maneuver. That this suite of controls (i.e., detection and recovery) has not been modeled is caught by the validation check that AdvoCATE performs—to detect event chains not having at least one barrier/control—and is highlighted on the BTD, as shown in Fig. 6.

Similarly, an example of a parallel split, in short, concerns the consequence event E7 of Fig. 1. In particular, this is a generic consequence but, in reality, it refers to a MAC event occurring *within* the OR. Since it is possible for a MAC event to occur in the situation where the UA exits the OR and then collides with external air traffic, it can be useful (and more accurate) to apply a parallel event split transformation to event E7, and associate the new event (and BTD) with the hazard H2.

E. Validations

Subtle problems with a safety system can be difficult to determine by manual inspection. In AdvoCATE, we have implemented a number of validations that inform the user when there are potential issues with the current state of the SA. Many of these check basic properties, such as the existence of controls between two events, as seen in Fig. 6, where the transformation resulted in a BTD that violates this condition. We briefly motivate two useful validations:

a) A key principle governing barrier/control usage is loose coupling and minimizing barrier interdependence to achieve risk reduction through diversity. This is violated when we repeat controls on a path, e.g., using the same control for prevention of a specific top event, as well as for recovery after that top event occurs. AdvoCATE warns against this repetition, by annotating the offending controls on the diagram.

b) In incrementally building up the safety architecture, paths can emerge that bypass or *short circuit* controls or barriers [21]. That is, when some controls/barriers on a path are breached, subsequent uses of the (different) controls belonging to the same barrier (on that path) are ineffective. In this case, AdvoCATE annotates the offending path.

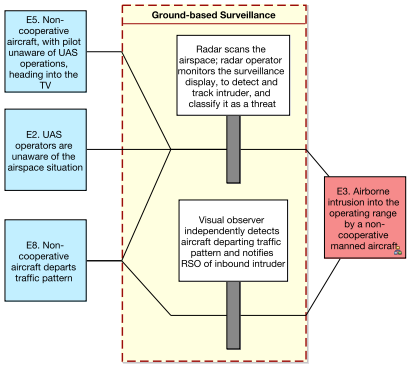


Fig. 7. Barrier-centric slice view for the *Ground-based Surveillance* barrier, being used in the safety architecture of the running example, as illustrated by the BTDs of Figs. 2 and 3

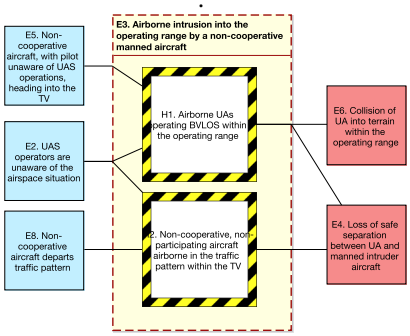


Fig. 8. Event slice view constructed across the safety architecture containing the BTDs of Figs. 2 and 3.

F. Views Supporting Comprehension

A useful consequence of creating a SA using our implementation in AdvoCATE, is the ability to automatically generate and update *views*, three of which we now describe, and that we believe have practical utility.

1) Barrier-centric Slice View: This view presents the controls associated with a given barrier alongside the threats and consequences that they mitigate. It can be considered as a high-level specification of the functionality to be delivered by a given barrier, and is relevant for those stakeholders who are responsible for (implementing or executing) the particular barrier and its respective controls.

Fig. 7 gives an example, as applied to the *ground-based surveillance* barrier, across the SA for the running example. As shown, it presents the different threats

that the barrier will manage, and the consequences that can occur upon barrier breach. Table II gives the basic view specification, which is populated with the results of the following AQL queries, using a $Barrier$ as the query root: *i*) $q_{BSControls}(r) = r.barrierControls$, which simply returns the controls within this barrier; *ii*) $q_{BSThreat}(r) = r.barrierInstances.controlInstances.controllingEvents().event.asSet()$, which navigates the model to call the $controllingEvents$ operation on every control instance in the barrier; and *iii*) $q_{threatL}(s) = s.consequenceControls().control$.

2) Event-slice view: The *event-slice view* presents all the hazards pertinent to a given system event, e.g., where there is a hazard release, along with all the threats and consequences of that event within the relevant situation. This view can be useful during the development and assessment of the overall risk analysis in focusing on a specific, high-priority/criticality event that occurs within different operating situations. The basic view specification for the event-slice (not shown) is similar to that of the barrier-centric slice of Table II. We omit the queries for this view, which are similar to the queries for the barrier slice view. Fig. 8 gives an example of an event-slice view as applied to the event E3 across the SA for the running example. As mentioned, this allows us to focus on a top event that is not localized to a given hazard, and provides the context of the event across the overarching collection of hazardous scenarios.

3) Crew Allocation View: The *crew allocation view* is a tabular presentation of the barriers that are allocated to specific crew members, serving to provide a direct mapping from crew roles to the specific responsibilities they will discharge. This type of view is operationally useful and the allocation from barriers (and the underlying controls) can help derive standard operating procedures, checklists of tasks, etc. The SA model enables controls to be annotated with specific roles (see Section III-B) and AdvoCATE provides the crew allocation view to enable those stakeholders to view the pertinent information. We provide a tabular representation (not given here), which summarizes the barriers that each mission role is allocated for mitigating each threat.

V. Concluding Remarks

In this paper, we described our framework for model-driven development of bow tie diagrams (BTDs), which we have leveraged to support the creation of a safety case required for regulatory approval to conduct pioneering NASA flight operations, involving a fleet of sUAS flying BVLOS. Specifically, we have developed an SA model that aggregates a collection of inter-related BTDs, providing a basis both for risk assessment, and for elaborating parts of the required safety justification. Our approach was motivated through practical challenges that arose during safety case development, including maintaining consistency amongst diverse safety artifacts, making principled changes to a safety architecture, and presenting the safety justification to the relevant stakeholders in such a way as to facilitate safety related decision making.

Indeed, the range of formal views and the supporting risk analysis were instrumental in successfully obtaining approval to fly. The BTDs of Figs. 1–3 are, in fact, abbreviated fragments drawn from much larger, more comprehensive BTDs which we created in the safety case corresponding to the real mission on which the running example has been based. In creating those BTDs, AdvoCATE automatically assembles the underlying SA transparently, maintaining consistency between the associated elements. Besides the views described in this paper, another view that we leveraged is the *barrier-centric* view [5], that abstracts the details of the controls and only presents the different barriers applicable for risk management in a specific scenario. This view was particularly useful in concisely communicating the risk reduction measures that were undertaken, thus providing assurance of operational safety in the presence of new surveillance and avoidance capabilities. That view also provided the basis for a rapid risk assessment.

In brief, the risk assessment involved computing the likelihood of occurrence of the top events and consequence events, from the initial probabilities of occurrence of the identified threats and the integrities of the barriers used. In general, the probability that an event occurs is computed using the *inclusion-exclusion principle*, from the probability of the disjunction over all paths leading to that event. That, in turn, relies on the computation of path probabilities which, for a path, is determined as the joint probability that all the events on that path (including barrier breaches) occur.

The approach and models described here represent only a first step towards our wider plans for a fully integrated approach to safety that will incorporate other components of safety cases [22]. For example, we have implemented navigation between elements of the safety architecture and safety arguments [4] that provide rationale and evidence for using specific barriers within the safety architecture, assumptions made in the risk analysis, properties of the overall SA as well as its constituent elements, e.g., sufficiency of the SA to cover the relevant threats, fitness of purpose of the barriers, etc.

As well as developing models to address other facets of the safety framework, we plan to implement functionality for the generation of other artifacts, such as crew checklists and task plans—an important, and potentially complex, element of an implemented safety system. AdvoCATE currently implements the pre-defined views we have described here, but we plan to develop a domain-specific query language to allow domain experts to define and customize views according to their needs.Furthermore, to ensure scalability, we plan to investigate incremental generation of views and running of validations.

We also see significant potential to support mission design, for example, by extending the tool's capabilities with features for impact and sensitivity analysis of a proposed safety system, possibly through temporal extensions [23]. Lastly, though our emphasis in this work has been on modeling *safety systems*, model-based techniques have traditionally been applied to the development and analysis of target systems [24], themselves, and we plan to investigate how our work can best be aligned with NASA's initiative in this area [25].

REFERENCES

[1] US Department of Transportation, Federal Aviation Administration (FAA), "Flight Standards Information Management System, Volume 16, Unmanned Aircraft Systems," Order 8900.1, Jun. 2014.

[2] FAA Air Traffic Organization, Safety and Technical Training Service Unit, "Transforming Risk Management: Understanding the Challenges of Safety Risk Measurement," https://go.usa.gov/xXxea, Dec. 2016.

[3] R. A. Clothier, B. P. Williams, and N. L. Fulton, "Structuring the safety case for unmanned aircraft system operations in non-segregated airspace," *Safety Science*, vol. 79, pp. 213 – 228, 2015.

[4] E. Denney and G. Pai, "Architecting a Safety Case for UAS Flight Operations," in *34th International System Safety Conference (ISSC)*, Aug. 2016.

[5] E. Denney, G. Pai, and I. Whiteside, "Modeling the Safety Architecture of UAS Flight Operations," in *Proceedings of the 36th International Conference on Computer Safety, Reliability, and Security (SAFECOMP 2017)*, in Lecture Notes in Computer Science (LNCS), S. Tonetta, E. Schoitsch, and F. Bitsch, Eds. Springer, Sep. 2017 (to appear).

[6] E. Denney and G. Pai, "Safety considerations for UAS ground-based detect and avoid," in *Proceedings of the 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC 2016)*, 2016, pp. 1–10.

[7] E. Denney and G. Pai, "Argument-based airworthiness assurance of small UAS," in *Proceedings of the 34th IEEE/AIAA Digital Avionics Systems Conference (DASC)*, Sep. 2015, pp. 5E4–1–5E4–17.

[8] R. Berthold, E. Denney, M. Fladeland, G. Pai, B. Storms, and M. Sumich, "Assuring ground-based detect and avoid for UAS operations," in *Proceedings of the 33rd IEEE/AIAA Digital Avionics Systems Conference (DASC)*, Oct. 2014, pp. 6A1–1–6A1–16.

[9] T. Prevot, J. Rios, P. Kopardekar, J. Robinson III, M. Johnson, and J. Jung, "UAS Traffic Management (UTM) Concept of Operations to Safely Enable Low Altitude Flight Operations," in *Proceedings of 16th AIAA Aviation Technology, Integration, and Operations Conference*, no. AIAA-2016-3292, Jun. 2016.

[10] E. Denney and G. Pai, "Tool support for assurance case development," *Automated Software Engineering*, 2017, to appear.

[11] FAA Air Traffic Organization, *Safety Management System Manual version 4.0*, Federal Aviation Administration, May 2014.

[12] C. Haddon-Cave, "The Nimrod Review: An independent review into the broader issues surrounding the loss of the RAF Nimrod MR2 Aircraft XV230 in Afghanistan in 2006," Report, The Stationery Office, London, UK, Oct. 2009.

[13] Adelard LLP. (2017). *Assurance and Safety Case Environment (ASCE) Software* [Online]. Available: http://www.adelard.com/asce

[14] CGE Risk Management Solutions. (2017). *BowTieXP Software* [Online]. Available: http://www.cgerisk.com/

[15] BowTie Pro. (2017). *BowTie Pro Software* [Online]. Available: http://www.bowtiepro.com/

[16] Meercat Pty Ltd. (2017). *Meercat RiskView Software* [Online]. Available: http://www.meercat.com.au/

[17] ABS Group. (2017). *THESIS BowTie Software* [Online]. Available: http://www.abs-group.com/,

[18] Nicestsolution. (2017). *SafetyBarrierManager Software* [Online]. Available: http://safetybarriermanager.com/

[19] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework 2.0*, 2nd ed. Addison-Wesley Professional, 2009.

[20] *Refactoring: Improving the Design of Existing Code*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[21] N. J. Duijm, "Safety-barrier diagrams as a safety management tool," *Reliability Engineering and System Safety*, vol. 94, no. 2, pp. 332–341, Feb. 2009.

[22] E. Denney and G. Pai, "Automating the Assembly of Aviation Safety Cases," *IEEE Transactions on Reliability*, vol. 63, no. 4, pp. 830–849, 2014.

[23] D. L. Mathias, S. Go, K. Gee, and S. Lawrence, "Simulation assisted risk assessment applied to launch vehicle conceptual design," in *Reliability and Maintainability Symposium*. IEEE, 2008.

[24] M. Güdemann and F. Ortmeier, "A framework for qualitative and quantitative formal model-based safety analysis," in *12th IEEE High Assurance Systems Engineering Symposium, HASE*, 2010, pp. 132–141.

[25] J. Evans, S. Cornford, and M. S. Feather, "Model based mission assurance MBMA: NASA's assurance future," in *Annual Reliability and Maintainability Symposium (RAMS 2016)*. IEEE, 2016, pp. 1–7.