# BayesianSafety - An Open-Source Package for Causality-Guided, Multi-model Safety Analysis

Robert Maier(✉) and Jürgen Mottok

Regensburg University of Applied Sciences, 93049 Regensburg, Germany
{robert.maier,juergen.mottok}@oth-regensburg.de

**Abstract.** Development and verification of modern, dependable automotive systems require appropriate modelling approaches. Classic automotive safety is described by the normative regulations ISO 26262, its relative ISO/PAS 21448, and their respective methodologies. In recent publications, an emerging demand to combine environmental influences, machine learning, or reasoning under uncertainty with standard-compliant analysis techniques can be noticed. Therefore, adapting established methods like FTA and proper tool support is necessary. We argue that Bayesian Networks (BNs) can be used as a central component to address and merge these demands. In this paper, we present our Open-Source Python package *BayesianSafety*. First, we review how BNs relate to data-driven methods, model-to-model transformations, and causal reasoning. Together with FTA and ETA, these models form the core functionality of our software. After describing currently implemented features and possibilities of combining individual modelling approaches, we provide an informal view of the tool's architecture and of the resulting software ecosystem. By comparing selected publicly available safety and reliability analysis libraries, we outline that many relevant methodologies yield specialized implementations. Finally, we show that there is a demand for a flexible, unifying analysis tool that allows researching system safety by using multi-model and multi-domain approaches.

**Keywords:** Fault Tree Analysis · Event Tree Analysis · Bayesian Networks · Causality · Package BayesianSafety

## 1 Introduction

Today's view on Functional Safety (FS) and reliability was developed over decades. Throughout this evolution, multiple techniques to manage different aspects of system safety emerged. Modern standards like IEC 61508 or its automotive relatives ISO 26262, ISO/PAS 26448, and UL 4600 encourage and support these modelling approaches. Consequently, methodologies like Fault Tree Analysis (FTA), Event Tree Analysis (ETA), Failure Mode and Effects Analysis (FMEA), or Goal Structuring Notation (GSN) form the basis that allows building complex, highly dependable [2] systems like autonomous driving cars.


M. Trapp et al. (Eds.): SAFECOMP 2022, LNCS 13414, pp. 17–30, 2022.
https://doi.org/10.1007/978-3-031-14835-4_2

Practitioners and researchers working with these modelling approaches demand appropriate tool support. Even though there are proprietary as well as Open-Source solutions available, most of them are optimized to work with one methodology only.

In recent years, researchers noted that system safety is a multi-aspect endeavour. Feth et al. [8] state that various of the above disciplines should be combined to form a joint safety engineering process in the automotive context. Similarly, Mosleh et al. [13] outline that different frameworks (here FTA, a modified ETA, and BNs) can be merged on a conceptual level. This allows addressing many requirements such as model maintainability and justifiability, incorporation of uncertainty, or high fidelity while staying compatible with various standards [18].

In the autonomous driving domain, research questions of interest include how reasoning under uncertainty, handling operational and environmental conditions, or a combination of abstract influences from multiple domains can be combined with ISO 26262 or ISO/PAS 21448. Modelling uncertainty is often addressed by resorting to BNs [12]. Moreover, Bayesian-based graphical models allow researching causality to answer questions like "what if" and "why" [15]. BNs are commonly used in conjunction with established, standard-compliant methods (e.g. Hybrid Causal Logic (HCL) [13]) and for model-to-model transformation [4,5,11].

Usually, the various FS and reliability methodologies have their own semantics, modelling assumptions, or mathematical frameworks to calculate metrics of safety evaluation like Average Probability of Failure on Demand or Probability of Failure per Hour. Due to this, software supporting these various methodologies differs drastically and renders a multi-method often a multi-tool approach. Additional features, like the incorporation of environmental aspects (e.g. adapting FTA as required by HCL) or combining various frameworks (e.g. Bow-Tie models (BTs)) are typically not supported by commercial tools like Ansys® medini analyze[1] or Open-Source packages like *SCRAM*[2]. Nonetheless, these capabilities are of high interest to researchers and practitioners alike. Ideally, software supporting the outlined combination of methodologies should be easy to adapt and modify, extensible, and foremost available for all.

The contribution of this paper and associated research is an Open-Source Python package called *BayesianSafety*[3], which can serve as a basic implementation to address the demands outlined above. Our aim is to provide a novel, extensible software environment with a focus on harmonizing the combination of various modelling approaches by using BNs.

First, we will cover the basics of BNs and how they can be used as a universal model to transform established standard-compliant methodologies. We will do so by addressing how our currently supported modelling approaches FTA and ETA can be mapped to BNs as a mathematical core framework. By doing so, we gain access to methods that can learn environmental models from data, combine them

[1] https://www.ansys.com/.
[2] https://github.com/rakhimov/scram.
[3] https://github.com/othr-las3/bayesiansafety.

with transformed models, and allow causal reasoning as outlined by [15]. Next, we give a brief overview of the resulting modelling possibilities by combining simple models and the currently implemented features of *BayesianSafety*. Based on requirements for a novel FS and reliability analysis software package, we review and compare related work. We close this paper with a summary as well as future research and implementation intentions.

## 2 Preliminaries

The following section highlights the key ideas behind our proposed software package *BayesianSafety*.

### 2.1 Bayesian Networks

Probabilistic Graphical Models (PGMs) like BNs are often used as suitable mathematical frameworks for reasoning under uncertainty [12]. BNs are directed acyclic graphs that are able to convey assumptions on how variables interact. They specify and represent a joint probability distribution and allow efficient computation of probabilistic information. When connections between variables are given a causal (i.e. cause and effect) interpretation, they can be used as causal models to facilitate causal reasoning [15].

Given the assumption that variables are only directly influenced by their Markovian parents $pa_i$ (i.e. immediate predecessor nodes), the underlying joint probability distribution $P(\mathbf{X})$ can be factorized as a special case of the chain rule of probability:

$$P(\mathbf{X}) = \prod_i P(x_i|pa_i) \tag{1}$$

A conditional probability distribution can be interpreted as a causal mechanism mapping the influence of parent nodes to the distribution of the child node. Causal models allow interventions (i.e. locally changing a causal mechanism) and estimating how probability distributions would change. These mechanisms can be used to model stochastic as well as deterministic relationships. This property is exploited in model-to-model transformations.

### 2.2 Model-to-Model Transformations

Bobbio et al. [4] show that Fault Trees (FTs) can be mapped into BNs without loss of expressiveness. The deterministic relationship between input nodes (e.g. basic events) and a node of interest (e.g. gate) can be modelled by adjusting the respective conditional probability distributions (i.e. implementing a truth table). In contrast to FTs, a straightforward topological transformation of Event Trees (ETs) [3] is usually not possible. Instead, the model's structure is defined by some properties of the paths between an initiating event and associated consequences. Both model transformations are reversible without any loss of information. A resulting

topology, together with the conditional distributions associated with each node, encode all required information of the original model in the resulting BN. To support reasoning under uncertainty, adapting formal methodologies like FMEA [11] or GSN [14] is also researched by using model-to-model transformations.

Casting various modelling approaches into the same mathematical framework allows merging them. In the case of BNs, this requires adjusting the causal mechanisms of nodes that link transformed models. This is possible as long as the modularity assumption (i.e. changing a local mechanism does not affect others, e.g. parents of a node) holds. Consequently, environmental models can be used to serve as input for standard-compliant approaches (e.g. rain and light conditions as basic events in a FT [18]).

Figure 1 gives an informal example of how (mathematically) independent frameworks can be transformed into BNs and combined to a single model. On the left side of the figure, a FT is mapped into a BN according to [4]. In the centre, an ET is transformed based on [3]. Combining FTs and ETs can be done via BTs [10]. On the right side, a HCL model is built from the individual parts. An environmental model with influences ($E_i$) is added and replaces two basic events of the FT via the nodes $E4$ and $E5$. The top-level event of the transformed FT is considered as initiating event of the ET with outcomes $FE1_{1a}$ and $FE1_{1b}$. FT, ET, and environmental influences together form an instantiation of the HCL framework. Additionally, the conditional probability tables for the transformed ET are given, showing the preservation of determinism after mapping.

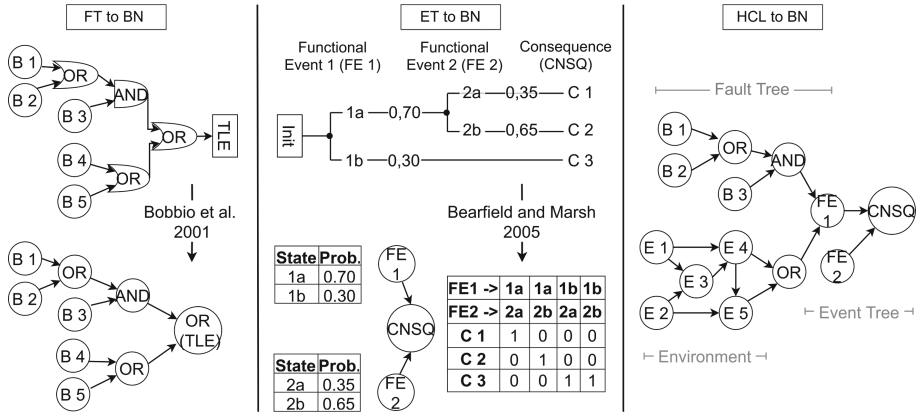


**Fig. 1.** Informal example of how a FT and an ET can be transformed into a BN. On the right side, both models are combined and extended by an environmental network, forming an instantiation of the HCL framework.

### 2.3 Bridging the Issue of Multiple Domains

Due to the increasing complexity of technical systems (e.g. autonomous driving cars) FS and reliability analysis is non-trivial. Components of a system

can be broadly categorized into three groups: software, hardware, or artificial intelligence-based. Depending on the individual category, different standards like ISO 26262 or ISO/PAS 21448, or UL 4600 might apply. Each of these standards encourages the use of different modelling approaches. As [8,16,18] among others point out, a combination of these methods can be reasonable.

A problem that arises when trying to merge methodologies, is how to adequately link them. Interactions between components as well as between model elements are often treated as causal. As mentioned above, BNs can be utilized to handle such relationships in the form of causal mechanisms and therefore qualify as a framework for a joint multi-domain, multi-model system evaluation.

With respect to the increasing popularity of Machine Learning (ML) based components, the issue of explainability emerges. This is due to black-box learning methods (e.g. Neural Networks) and the data used for their training.

A trend called "causal revolution" started recently in the ML community. Its core idea is to employ causal knowledge in the form of causal graphs to train learning algorithms more efficiently and robustly [17]. This allows working with white-box models (i.e. causal graphs). In the context of FS and reliability, they can serve as input to standard-compliant methodologies like FTs and may be used to verify parts of an ML algorithm.

Causal models can also be used to interpret the underlying data and encoded relationships (i.e. correlation and causation) between variables [15]. Causal discovery describes the approach to algorithmically learn causal models from data, by estimating the topology and conditional probability distributions of the graph [19]. This is especially valuable in light of data-driven safety assurance, as it allows the processing of collected data as a source of causal knowledge (i.e. as an environmental model). As a consequence, the modelling of environmental, operational, or scenario-relevant influences can be decoupled in parts from human experts. Therefore, causal models might serve as an objective way to address parts of scenario-based testing approaches (e.g. evaluation of sensor data or simulation results) as outlined in ISO/PAS 21448.

As mentioned in Sect. 1, each modelling approach typically builds on different assumptions. This ties the calculation of relevant metrics to specialized mathematical constructs. For example, FTs are based on boolean logic and can be used to calculate the likelihood of component failure (e.g. top-level event) given influencing factors. Based on boolean algebra or via a Binary Decision Diagram (BDD), risk worth or importance measures of a component can be derived. ETs on the other hand model logical combinations of events that lead to different consequences. Corresponding likelihoods are calculated as a product of branching probabilities based on paths between an initiating event and a consequence.

These different paradigms yield tailored and highly customized software packages. Researching combinations of these methodologies or of potential environmental influences leads to a multi-tool endeavour. Even though combining different domains is reasonable, to the best of our knowledge, currently, no Open-Source tool supports it. Our contribution is intended to address this demand.

## 3 Package BayesianSafety

In the following, we will describe our Python package *BayesianSafety*. The descriptions given below refer to FTs as an example.

### 3.1 Models and Their Combinations

As described by [13] a suitable combination of methodologies can jointly address various demands of system safety. Looking at Probabilistic Safety Assessments (PSAs) the two main frameworks used are FTA and ETA. Both analysis methods can be combined and transformed into BNs and extended in different ways (see Sect. 2). Due to this fact, they were chosen as the core functionality of *BayesianSafety*. Description and exchange of classic PSA models are supported by the Open-PSA initiative model exchange format [7]. Relevant combinations of the above methodologies include:

- stand-alone "classic" FT or ET,
- ET and BN (environmental influences) (see e.g. [3]),
- FT and BN (environmental influences) or common causes (see e.g. [4]),
- FT and ET (BT, see [10]), or
- FT and ET and BN (environmental influences) (HCL, see e.g. [13,18]).

*BayesianSafety* was developed to allow specifying models by hand if no description file can be provided. Listing 1.1 shows the code to set up the FT model of Fig. 1. Probability nodes (i.e. basic events) have a name parameter, can be given a static probability of failure, and are assumed to have two states: working and failing. A flag can be set to indicate time dependency. The probability of failure is then treated as failure rate $\lambda$ for a default time behaviour of $1 - e^{(-\lambda t)}$. Logic gates (i.e. boolean gates) are given a name, a list of parent nodes, and a logic type to connect these arbitrary inputs.

**Listing 1.1.** Example listing for defining a simple FT in *BayesianSafety*.

```
1  from bayesianfaulttree.FaultTreeProbNode import
       FaultTreeProbNode as FTProb
2  from bayesianfaulttree.FaultTreeLogicNode import
       FaultTreeLogicNode as FTLogic
3  from bayesianfaulttree.BayesianFaultTree import
       BayesianFaultTree as BFT
4
5  B_1 = FTProb('B_1', 0.5e-3)
6  B_2 = FTProb('B_2', 1.6e-3)
7  B_3 = FTProb('B_3', 2.7e-3)
8  B_4 = FTProb('B_4', 3.8e-3)
9  B_5 = FTProb('B_5', 4.9e-3, is_time_dependent=True)
10
11 OR_1 = FTLogic('OR_1', ['B_1', 'B_2'] , 'OR')
```

```
AND  = FTLogic('AND', ['OR_1', 'B_3'], 'AND')
OR_2 = FTLogic('OR_2',  ['B_4', 'B_5'], 'OR')
TLE  = FTLogic('TLE', ['AND', 'OR_2'], 'OR')

probability_nodes = [B_1, B_2, B_3, B_4, B_5]
logic_nodes = [OR_1, AND, OR_2, TLE]
model = BFT("Example", probability_nodes, logic_nodes)
```

FTs and ETs can be imported either as individual models or as a joint model in case of a BT if a suitable Open-PSA file is available. An importer parses the provided tree structures and preprocesses relevant information. Based on the model type, a mapper is invoked. It instantiates a BN that serves as the default internal model-class of *BayesianSafety*. A *networkX DiGraph*-object [9] is used as a container for the graph representation of the BN topology. Mapping all imported networks to a BN allows combining models and joint inference. Listing 1.2 shows how a simple ET and FT can be imported.

**Listing 1.2.** Example listing for loading an ET and a FT from an Open-PSA file.

```
from bayesianeventtree.EventTreeImporter import
    EventTreeImporter
from bayesianfaulttree.FaultTreeImporter import
    FaultTreeImporter

bay_FT = FaultTreeImporter().load('./Example.xml')
bay_ET = EventTreeImporter().load('./Example.xml')
```

**Working with Fault Trees.** FTs can be used either as a quantitative or qualitative representation of a modelled system. Some basic metrics of interest include minimal cut sets and different importance measures (e.g. Risk Reduction Worth, Risk Achievement Worth, or Birnbaum importance). *BayesianSafety* supports all of the above, including the ability to run a time simulation between a start and end time. For a time-dependent evaluation, the respective time behaviour of time-dependent nodes is evaluated at each time step and the resulting probability of failure is updated for each affected element in the BN. For each node of the tree, the evolution of the probability of failure can be plotted or saved as a figure. A FT can also be evaluated at a given mission time $t$.

All probability evaluations are done by inferring the BN (i.e. the transformed model). It should be noted, that some method-specific results like nodes contributing to a minimal cut set are currently calculated based on the original FT structure (via the MOCUS algorithm) if equivalent methods using a BN are not available.

A missing feature in most Open-Source packages is the ability to freely specify a custom time behaviour for basic events. The default assumption for an underlying reliability function is usually to be exponential. This holds during the system's lifetime, but neglects for example end-of-life effects. *BayesianSafety*

allows modelling time dependency for any probability node of a FT model by specifying a custom function. Listing 1.3 gives a short example of how a full network evaluation including the definition of a custom time behaviour can be implemented.

**Listing 1.3.** Example listing for evaluating a FT where two nodes have a customized time behaviour (i.e. sigmoid and cosine).

```
import numpy as np
def time_fn(time, kind="cos"):
  if time <= 0:
    return 0
  sig = 1 - 1 / (np.exp(1.23e-4 * time) + 1)
  return np.cos(time) if kind == "cos" else sig

ft_model = ...
node_1 = ft_model.get_elem_by_name("target_node_1")
node_1.change_time_behaviour(time_fn, {"kind":"sigm"})
node_2 = ft_model.get_elem_by_name("target_node_2")
node_2.change_time_behaviour(time_fn)

ft_model.run_time_simulation(start_time=0, stop_time=1e5,
    simulation_steps=50, plot_simulation=True)
```

**Extension by Linking Environmental Models.** Since FTs are internally represented as BNs, combining them with environmental models can be done straightforward. Boolean gates in a FT are interpreted as potential mounting positions that can be extended by target nodes of one or multiple environmental models. Additional mapping information needs to be provided to define links between gates and external variables. In a resulting *extended* FT, environmental influences are treated as new binary basic events, allowing any calculation of the metrics mentioned above. Since environmental variables can have an arbitrary number of states (e.g. weather with states rain, fog, and snow), one of them needs to be selected and will be treated as "failing". An associated probability is then interpreted as a static probability of failure. Figure 2 shows the idea of linking a FT with multiple environmental models.

Environmental influences may not only serve as basic events. Depending on the modelled effects (e.g. occlusion of a camera lens due to precipitation), an underlying assumption about the failure behaviour of a component (e.g. camera) may change. In *BayesianSafety* this can be implemented by treating an environmental node as a trigger and in response modifying an *existing* basic event based on external influences. Consequently, a time-independent node can be given a custom time behaviour. A modification changes the static probability of failure or the default reliability function $R(t) = e^{(-\lambda t)}$ of a node based on selected state probabilities $P(env_i)$ of environmental nodes and predefined thresholds. Note

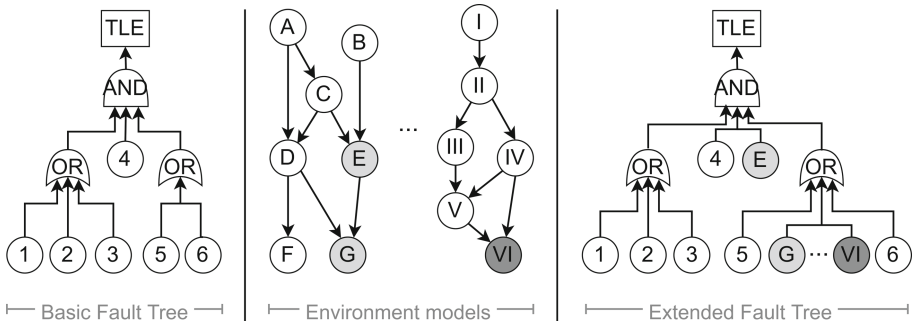


**Fig. 2.** A FT can be combined with multiple environmental models. Adequate mapping information needs to be provided, specifying which gates will be extended. Selected environmental nodes are treated as new basic events in a resulting *extended* FT.

that nodes independent of external influences can still be modified as described above. The following alterations are currently supported:

**Replacement:** $R(t) \rightarrow P(env)$
**Addition:** $R(t) \rightarrow \omega_0 R(t) + \omega_{env} P(env)$
**Weighting:** $R(t) \rightarrow \omega_0 R(t) \prod_i \omega_i P(env_i)$
**Rate:** $R(t, \lambda) \rightarrow R(t, \lambda^\star)$
**Parametric:** $R(t, \lambda) \rightarrow R^\star(t, P(env))$ as a special case of "Rate"
**Functional:** $R(t, \lambda) \rightarrow R^\star(t, \mathbf{X})$ where $\mathbf{X}$ is a set of parameters.

### 3.2 Model Inference

Typical metrics calculated in FTA and ETA represent prior probabilities. Posterior distributions (e.g. $P(X|Y, W)$) can be calculated easily due to the use of BNs as a mathematical core framework.

In *BayesianSafety*, inference is tied to a single, independent BN. Queries and their results are only computed on that network instance. In combined models, this leads to a problem for some evaluations due to the current implementation. Suppose we want to extend a FT by linking environmental influences as described above. The resulting composite BN is a newly instantiated model, containing copies of all nodes of the FT and new binary basic events based on the specified environmental nodes. A problematic query of interest would be, how an observed top-level event in the FT part affects the posterior probability for an environmental node $Weather$ (e.g. $P(Weather|TLE = failure)$). Even if $Weather$ is formally linked to the FT, a distributional update will only consider the newly created basic event node $Weather_{new} = state\ x$ in the composite BN and not the original node in the environment network. Consequently, this means that the environmental model will not be considered at all. If an update is required, the joint model needs to be created accordingly.

*BayesianSafety* is developed to support causal inference based on BNs. To do this, routines for handling interventional (e.g. $P(X|do(Y), W)$) and counterfactual (e.g. $P(y_{x'}|x, y)$) queries as described by [15] are available. For example, interventional queries can be used to evaluate the effects of a forced event. Imagine a redundant architecture consisting of two independent sub-systems. $P(X|do(sub_1 = fail))$ describes how actively disabling *sub-system 1* influences a component $X$ and therefore partially evaluates the effectiveness of *sub-system 2*. In FTA, this relates to modelling the respective branches with a house event which is set to true or false respectively.

Causal inference allows answering interventional questions like "What if component $X$ would fail" and counterfactuals like "Would rain have caused $X$ to fail, given we know it was sunny and $X$ worked". This may especially be relevant for generating insights for environmental models by researching causal influence among variables.

### 3.3 Technical Ecosystem

*BayesianSafety* is developed in Python 3.9+ and currently spans around 6500 lines of code with an average cyclomatic complexity of 4.76. Source code, including examples treating FTA and ETA, can be found in our GitHub repository under https://github.com/othr-las3/bayesiansafety.

Instead of implementing inference algorithms ourselves, we rely on two different computational back-ends, namely *pgmpy* [1] in version 0.1.17+ and *pyAgrum* [6] in version 0.22.5+. Both provide a wide variety of approximate and exact inference and structure learning algorithms, are actively developed, and are established in the Open-Source PGM software community. They enable causal inference and support other PGM families like Markov Networks.

A key feature of both packages is state-of-the-art causal discovery methods. Consequently, this renders our package ready for an extension to provide an end-to-end (i.e. data to insight) capability. Environmental models could then be learned and combined with standard-compliant approaches to system safety, as outlined throughout this paper. Relying on third-party back-ends allows focusing on the implementation of required methodologies. BNs serve as generic containers to run all probability calculations on. As long as a modelling approach can be cast into the formalisms of BNs, it is expected to be implementable in *BayesianSafety* with minimal effort.

Parsed trees, as well as the underlying graph structure of a model, are internally represented as *networkX DiGraph*-objects [9]. *NetworkX* is one of the richest libraries for managing graphs in Python and provides a vast amount of graph algorithms. It allows plotting graphs, has export and import capabilities to different graph exchange formats, and allows adding custom data to graph elements. Figure 3 gives an informal overview of the ecosystem and the architectural idea of *BayesianSafety* described above.

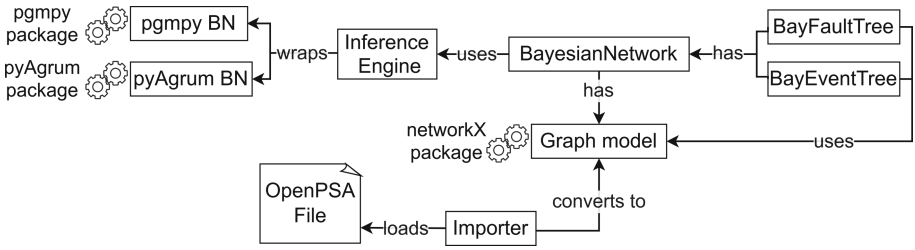


**Fig. 3.** During inference, *BayesianSafety* acts as a wrapper for back-end packages *pgmpy* [1] and *pyAgrum* [6]. Graph representations are implemented as *networkX DiGraph*-objects [9] and can be loaded from Open-PSA model exchange format files. FS and reliability methodologies like FTA or ETA and their respective algorithms can be implemented as custom modules building on BNs as a core framework.

## 4 Related Work

Zurheide et al. [20] recently developed a Python package called *pyBNBowtie*[4] to work with BTs mapped to BNs. They support the Open-PSA exchange format[5] to provide models and cast them directly into a *pgmpy BayesianNetwork*-object. Technically, they build on the model-to-model transformations described in [3, 4, 10] but support only parts of the arc simplifications described by [3]. Features like Minimal-Cutset calculation in FTs are not implemented.

Open-PSA and therefore treatment and evaluation of individual FTs and ETs are also supported by the C++-based library *SCRAM*. The R package *FaultTree*[6] focuses on FTA, partially builds on *SCRAM*, and adds a basic graphical user interface. Both use BDDs for the calculation of gate probabilities and efficient inference of large models.

*JReliability*[7] is a Java-based package that also uses BDDs to model trees that are connected via boolean functions. It supports calculating different reliability metrics like Birnbaum importance, Risk Reduction Worth, or Mean-Time-To-Failure, and allows the visualization of metrics and distributions over time.

*meta4ics*[8] is another Java-based tool for generic AND/OR-connected graphs and can be used to identify critical nodes by calculating a custom weighting metric. *SCRAM*, *JReliability*, and *FaultTree* are highly optimized to work with FTs and support additional boolean gate types other than AND/OR. All of these tools lack the capability to support emerging demands as described in Sect. 1.

Most publicly available software packages for risk assessment and reliability evaluation address FTA and ETA. They are typically monolithic, rendering extension non-trivial. Due to various languages or visualization capabilities used,

[4] https://github.com/zurheide/pybnbowtie.
[5] https://open-psa.github.io/joomla1.5/index.php.html.
[6] https://github.com/jto888/FaultTree/.
[7] https://github.com/SDARG/jreliability.
[8] https://github.com/mbarrere/meta4ics.

they are mostly dependent on a specific platform. Adapting them to work with other modelling approaches is not feasible. Based on the scope of this paper and available related work, the following properties for a modern Open-Source FS and reliability software can be derived:

**Cross-platform.** The used programming language and associated third-party libraries for implementing algorithms should be independent of a computing platform to address a wide range of hardware and users.
**Exchangable.** Different algorithmic libraries (back-ends) should be available to access required features with minimal changes to the package code.
**Exact.** Implemented methodologies produce the same results as their classic (e.g. BDD) implementations.
**Extensible.** Implementing custom functionalities should be possible with low effort if they can be cast to a common mathematical framework (e.g. BNs).
**Uncertainty.** The central mathematical framework used should allow reasoning under uncertainty and modelling of deterministic relationships.
**Modular.** Standard-compliant methodologies can be used stand-alone or combined with other methods or environmental models.
**Data oriented.** ML learning approaches, as well as treatment of environmental data, should be possible in the same tool.
**Causal.** Support of causal inference to facilitate causal reasoning.

The above Open-Source packages are not intended to incorporate environmental influences. A combination of models as described throughout Sect. 2 is not possible. They lack any support for causal inference, with the only exception being *pyBNBowtie* due to it using *pgmpy*. Neither do they have the option to be extended by data-driven approaches.

As generic, multi-purpose PGM libraries, *pgmpy* and *pyAgrum* support modelling, learning, and inference of BNs and other of the above requirements. Therefore, they qualify as candidates to bridge multiple domains at the cost of implementing standard-compliant methodologies by hand.

In the light of the recent trend of incorporating uncertainty and the demand to consider environmental influences (e.g. scenarios as suggested by ISO/PAS 21448) all of the above packages lack some functionality. As a consequence, multi-methodology approaches as encouraged by [8,18] and others require the use of multiple software tools. Our proposed package *BayesianSafety* is the first to address all of the above requirements.

## 5 Conclusion

Bayesian Networks are an established framework to deal with uncertainty. In the light of researching system safety, they have desirable properties like versatility, comprehensibility, and support of causal reasoning. In recent years, multiple model-to-model transformations from classic analysis methodologies into BNs have been researched. Many of these publications state that addressing causality is feasible by using PGMs. Since BNs are agnostic to *what* they model,

dealing with environmental, socio-technical, or abstract influences, is possible. This enables the combination of models with different scopes and formalisms.

In this paper, we outlined some emerging areas of interest in the FS and reliability community (e.g. researching environmental influences and their effects on reliability functions). Currently, no Open-Source software package is available that satisfies desirable properties like the support of causal reasoning, the ability to reason under uncertainty, or a combination of multi-domain models. To help address this, we propose our Open-Source Python package *BayesianSafety*. To the best of the authors knowledge, *BayesianSafety* is the first attempt to focus on the above-listed requirements and is intended to fill the demand for a causal, multi-domain, multi-model, analysis tool.

Our goal is to create a software environment, where common analysis methods can be treated together, by harmonizing the way each is processed mathematically. Due to the early implementation stage, only essential functionality to work with FTs, ETs, and environmental models is available. We plan on extending the provided capabilities as well as adding support to work with environmental models based on data. We hope *BayesianSafety* can serve as a baseline implementation for researching the combination of methodologies and encourage causality-guided system safety.

**Acknowledgment.** The present paper is supported by *Bayerisches Staatsministerium für Wirtschaft, Landesentwicklung und Energie* through the granting of the funding project *HolmeS*$^3$ (FKZ: DIK0173/03). We thank L. Grabinger and D. Urlhart for valuable discussions.

## References

1. Ankan, A., Panda, A.: pgmpy: probabilistic graphical models using Python. In: Proceedings of the 14th Python in Science Conference (SCIPY 2015). Citeseer (2015)
2. Avizienis, A., Laprie, J.C., Randell, B.: Fundamental concepts of dependability. Technical report series. Department of Computing Science (2001)
3. Bearfield, G., Marsh, W.: Generalising event trees using Bayesian networks with a case study of train derailment. In: Winther, R., Gran, B.A., Dahll, G. (eds.) SAFECOMP 2005. LNCS, vol. 3688, pp. 52–66. Springer, Heidelberg (2005). https://doi.org/10.1007/11563228_5
4. Bobbio, A., Portinale, L., Minichino, M., Ciancamerla, E.: Improving the analysis of dependable systems by mapping fault trees into Bayesian networks. Reliab. Eng. Syst. Saf. **71**(3), 249–260 (2001). https://doi.org/10.1016/S0951-8320(00)00077-6
5. Cai, B., Liu, Y., Liu, Z., Chang, Y., Jiang, L.: Bayesian Networks for Reliability Engineering. Springer, Singapore (2020). https://doi.org/10.1007/978-981-13-6516-4
6. Ducamp, G., Gonzales, C., Wuillemin, P.H.: aGrUM/pyAgrum: a toolbox to build models and algorithms for Probabilistic Graphical Models in Python. In: 10th International Conference on Probabilistic Graphical Models. Proceedings of Machine Learning Research, Skørping, Denmark, vol. 138, pp. 609–612, September 2020. https://hal.archives-ouvertes.fr/hal-03135721

7. Epstein, S., Rauzy, A., Reinhart, F.: The open PSA initiative for next generation probabilistic safety assessment. Kerntechnik **74**, 101–105 (2009). https://doi.org/10.3139/124.110020
8. Feth, P., et al.: Multi-aspect safety engineering for highly automated driving. In: Gallina, B., Skavhaug, A., Bitsch, F. (eds.) SAFECOMP 2018. LNCS, vol. 11093, pp. 59–72. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99130-6_5
9. Hagberg, A., Swart, P., Chult, D.S.: Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Lab. (LANL), Los Alamos, NM, United States (2008)
10. Khakzad, N., Khan, F., Amyotte, P.: Dynamic safety analysis of process systems by mapping bow-tie into Bayesian network. Process Saf. Environ. Prot. **91**(1), 46–53 (2013). https://doi.org/10.1016/j.psep.2012.01.005
11. Kirchhof, M., Haas, K., Kornas, T., Thiede, S., Hirz, M., Herrmann, C.: Root cause analysis in lithium-ion battery production with FMEA-based large-scale Bayesian network. arXiv:2006.03610 [stat], June 2020. https://doi.org/10.20944/preprints202012.0312.v1
12. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. Adaptive Computation and Machine Learning, MIT Press, Cambridge (2009)
13. Mosleh, A., Dias, A., Eghbali, G., Fazen, K.: An integrated framework for identification, classification, and assessment of aviation systems hazards. In: Spitzer, C., Schmocker, U., Dang, V.N. (eds.) Probabilistic Safety Assessment and Management, pp. 2384–2390. Springer, London (2004). https://doi.org/10.1007/978-0-85729-410-4_383
14. Nešić, D., Nyberg, M., Gallina, B.: A probabilistic model of belief in safety cases. Saf. Sci. **138**, 105187 (2021). https://doi.org/10.1016/j.ssci.2021.105187
15. Pearl, J.: Causality: Models, Reasoning and Inference, 2nd edn. Cambridge University Press, Cambridge (2009)
16. Rudolph, A., Voget, S., Mottok, J.: A consistent safety case argumentation for artificial intelligence in safety related automotive systems. In: ERTS 2018: 9th European Congress on Embedded Real Time Software and Systems (ERTS 2018), Toulouse, France, January 2018
17. Schölkopf, B., et al.: Toward causal representation learning. Proc. IEEE **109**, 612–634 (2021). http://arxiv.org/abs/2102.11107
18. Thomas, S., Groth, K.: Toward a hybrid causal framework for autonomous vehicle safety analysis. Proc. Inst. Mech. Eng. Part O J. Risk Reliab. (2021). https://doi.org/10.1177/1748006X211043310
19. Vowels, M.J., Camgöz, N.C., Bowden, R.: D'ya like DAGs? A survey on structure learning and causal discovery. CoRR abs/2103.02582 (2021). https://arxiv.org/abs/2103.02582
20. Zurheide, F.T., Hermann, E., Lampesberger, H.: pyBNBowTie: Python library for bow-tie analysis based on Bayesian networks. Procedia Comput. Sci. **180**, 344–351 (2021). https://doi.org/10.1016/j.procs.2021.01.172. Proceedings of the 2nd International Conference on Industry 4.0 and Smart Manufacturing (ISM 2020)