SGSN: Tool Support for GSN Development

Radouane Bouchekir, Fatiha Ouazar, Mohand Cherif Boukala, and Malika Ioualalen

MOVEP, Computer Science Faculty
USTHB
BP 32 El-Alia, Algiers, Algeria
{rbouchekir, fouazar, mboukala, mioualalen}@usthb.dz

Abstract. Graphical argument structures such as Goal Structuring Notation (GSN) are increasingly being used in practice as a means to provide safety assurance to stakeholders, such as regulatory authorities, regarding the dependability and safety of a system. These argument-based assurance cases are utilized in practice to demonstrate that a system is suitable for its intended use. In this paper, we present Structured GSN (SGSN), an open-source tool to create and edit assurance cases. In addition to basic functionalities related to the creation of GSN, we added two types of analysis, syntactic analysis which checks if the developed GSN satisfies structured criteria, and semantic analysis which checks compartmental properties. For the semantic analysis, we mapped the GSN to a formal model i.e., Markov Decision Process (MDP), and checked properties expressed in PCTL using a probabilistic model-checker.

Keywords: Safety assurance, Goal Structuring Notation (GSN), Markov Decision Process.

1 Introduction

1.1 Context

Certification of safety-critical systems is a rigorous evaluation process to certify that these systems meet specific safety and reliability standards [1], [2]. It involves in-depth analyses, tests, and assessments carried out by independent bodies or regulatory authorities. The aim is to guarantee compliance, minimize risk, and promote stakeholder confidence in the operation of complex and safety-critical systems.

Safety assurance [3], [4] is a process used to assess and demonstrate that complex systems comply with safety and security requirements. This includes the identification and analysis of risks, the definition of appropriate safety measures, as well as validation and verification of safe performance. One of the motivations for using safety assurance is to explicitly capture the traceability between safety claims and the substantiating evidence and this simplifies the certification process. Another motivation is to make it easier to understand and critically review the safety assurance to further improve clarity.

In practice, safety assurance can be represented in different formats, such as: *text*, *table*, or graphical notations like *ACE (Argumentation, Claims, and Evidence)* [5], and *GSN (Goal Structuring Notation)* [6]. The choice of representation can depend on factors such as the audience, the complexity of the safety assurance, and the level of detail required for communication and understanding. Indeed, representation as text and table are not suitable for the safety assurance of complex systems due to their ambiguity. In such cases, graphical representation using ACE or GSN can give a clearer and more comprehensive representation. Furthermore, graphical notations have emerged over the past decade to present the elements of a safety argument, e.g., CAE and GSN. Such graphical argument structures can be thought of as a visual index into the safety reasoning and evidence comprising a safety assurance.

In this paper, we consider GSN as the graphical notation to represent safety assurance. GSN allows representing in a clear and structured way the safety objectives, the evidence

needed to achieve them, along with the arguments and justifications that support this evidence. it consists of a network of interconnected elements, including *goals*, *claims*, *evidence*, and *assumptions*. Safety assurance can be represented in GSN by creating a goal node that represents the overarching safety objective. Claims and evidence nodes can be added to support the goal, with arguments and assumptions linking them together. This visual representation helps to show the logical structure of the safety assurance argument and provides a clear framework for understanding the supporting evidence.

1.2 Contributions

GSN plays an important role in the safety assurance process, it facilitates communication and understanding of safety assurance activities between stakeholders involved in the certification of safety-critical systems. However, the lack of tools supporting the creation and analysis of GSN makes the use of this notation challenging. The main challenges can be summarized as: (i) The lack of tools to create and maintain GSN structure, and (ii) The need to perform analysis (syntactic and semantic) to ensure the correctness and coherence of created GSN.

In this work, we present our tool Structured GSN (`SGSN` [1]), where we aim to address the above-mentioned challenges. Our main contributions are:

- Implementation of an open-source tool to facilitate the creation and maintenance of GSNs,
- Syntactic analysis to ensure the coherence of GSN w.r.t criteria described in the latest standard [6],
- Semantic analysis to verify compartmental properties.

The syntactical analysis aims to check the conformity of the GSN diagrams to the notation rules described in the current standard [6], while semantic analysis aims to verify the consistency and validity of the relationships and arguments between the different elements of the GSN. To ensure the semantic analysis, we proposed to formally specify GSN models as MDP (Markov Decision Process) [7] and then check the validity of formal properties expressed in the Probabilistic Computation Tree Logic (PCTL) [8] formula. This ensures verifying the logical consistency and validity of arguments present in GSN diagrams w.r.t the specified property.

The rest of this paper is structured as follows, section 2 summarizes the preliminaries used to illustrate our contributions. In section 3 we illustrate the SGSN meta-model. Sections 4 and 5 illustrate the proposed syntactic and semantic analysis, respectively. Section 6 highlights related works. Finally, we provide concluding remarks and pointers to future work in Section 7.

2 Preliminaries

2.1 Safety assurance & GSN

Safety Assurance is a comprehensive, defensible, and valid justification of the safety of a system for a given application in a defined operating environment [3], [4]. The main aspect of this justification, which is a risk-based argument, is to link safety claims through

[1] `https://github.com/BouchekirRedouane/SGSN`

a reasonable chain to the set of evidence that can support these claims.

The GSN standard defines the assurance case as *"a reasoned and compelling argument, supported by a body of evidence, that a system, service or organization will operate as intended for a defined application in a defined environment"* [6]. Figure 1 shows the core elements of GSN. In summary, the graphical symbols are as follows:

- Claim (known as a *Goal* in GSN-speak) represented as a rectangle,
- Strategy, represented as a parallelogram contains, that shows what strategy is used to justify the claim,
- Context, represented as a rounded rectangle contains. Usually, the claim is associated and valid in a particular context,
- Assumption represented as an oval contains,
- Solution represented as a circle, which is a description of the evidence that will be presented to justify the claim,
- Incomplete arguments are represented as a diamond represents.

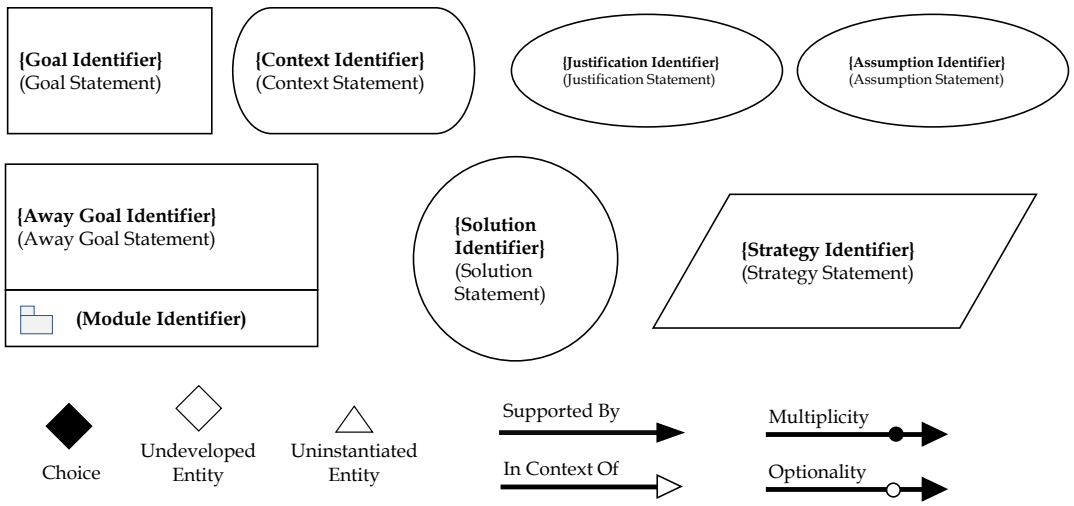

Fig. 1. Core elements of GSN.

2.2 Probabilistic Model

Markov Decision Processes (MDP) [7] are frequently employed to model and analyze systems characterized by non-deterministic and stochastic behavior.

Definition 1. *An MDP is a tuple* $M = (S_M, s^0_M, \Sigma_M, \delta_M)$*, where* S_M *is a finite set of states,* s^0_M *is an initial state,* Σ_M *is a finite set of actions and* $\delta_M \subseteq S \times (\Sigma_M \cup \{\tau\}) \times Dist(S)$ *is a probabilistic transition relation.*

In a state s of MDP M, one or more transitions, denoted $(s, a) \rightarrow \mu$, are available, where a is an action label, μ is a probabilistic distribution over states and $(s, a, \mu) \in \delta_M$.

In this work, we focus on verifying probabilistic properties specified using Probabilistic Computation Tree Logic (PCTL) [8] in the form of $P_{\leq \rho}[\psi]$ with $\psi \in [0, 1]$ and

$$\phi ::= true | a | \phi \wedge \phi | \neg \phi$$
$$\psi ::= \phi \cup \phi$$

3 SGSN Meta-Model

A (non-modular) argument structure in GSN is illustrated in Fig. 2. It contains a top-level (root) goal stating a safety claim, which is in this example *" Safe_UAV_1: The Temporal Planning Network (TPN) solution finder should ensure that, for every generated route mission, this route should not violate geo-fences (NFZ)"*.

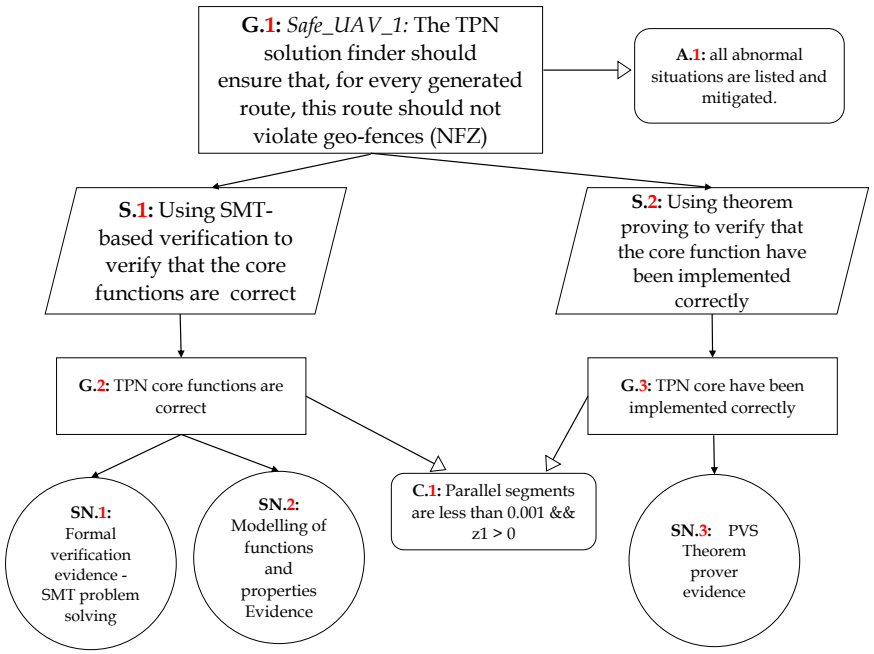

Fig. 2. A simple example of GSN used to illustrate that the AI-based planning for Unmanned Aerial Vehicles (UAV) is safe w.r.t to geo-fences constraints [9].

This goal is established under the assumption that *"all abnormal situations are listed and mitigated."*. The top-level goal is developed using two strategies, the first strategy is to verify the goal using an SMT-based verification technique, while the second is based on theorem proving. The sub-goals claim that the TPN core functions are correct (design) and implemented correctly (code). The solutions are the leaves of the GSN, and they represent the assurance evidence to support the sub-goals. The goals, strategies, and solutions are connected using the *"Is Supported by"* connectors, while context, assumption, and justification elements require an *"In Context Of"* connectors. GSN also employs a graphical annotation ($\Diamond$) for goals and strategies to signify that they are to be developed (tbd), implying they are incomplete. The top level of any safety assurance model based on the `SGSN` meta-model (Fig. 3) is a safety assurance represented as GSN. Essentially, this is the container that holds all GSN elements of the safety assurance i.e. GSN.

The GSN elements can be concrete instances of either an argument element or a relation element. An argument element generalizes the different types of GSN elements, i.e., Goal, Strategy, Assumption, Justification, Context, and Solution[2].

The attributes of an argument element are:

– *Identifier.* it contains a unique name for each element.

[2] Strictly speaking, the GSN uses the term Solution. We use the term Evidence to illustrate the artifacts attached to a solution as assurance evidence. for example files or external links.

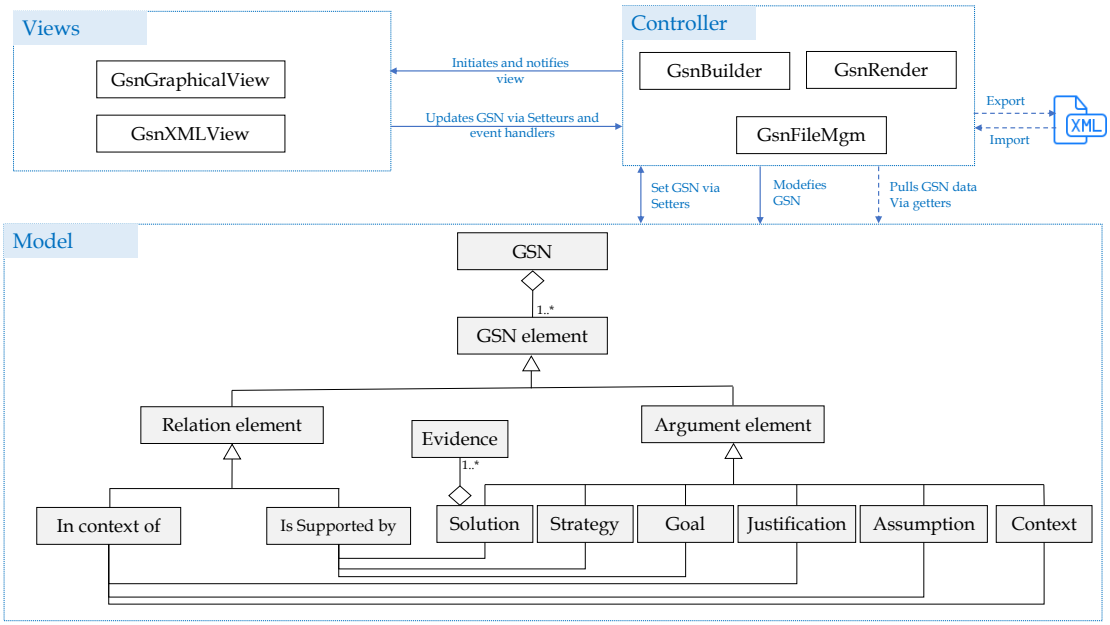

Fig. 3. Meta-Model of GSN Elements considered in our tool SGSN.

– *Description.* a describing associated with the argument element.
– *Attributes.* which are used to hold extra meta-data about the argument element.

In addition, we based our conceptual architecture on the MVC (Model-View-Controller) pattern. This patern helps in organizing and structuring the codebase, making it more maintainable, scalable, and easier to understand.

4 Syntactic Analysis

Syntactic analysis of GSN is the process of validating the structure of a GSN based on the current GSN standard (Version 3) [6]. The aim of syntactic analysis is to ensure that the GSN model is well-formed and respects the specific syntax rules, in order to obtain a clear and accurate representation of the safety assurance. Let first describe formally a safety assurance.

Definition 2. *We define an argument structure A, as a tuple $\langle N, l, \rightarrow \rangle$, where:*

– *N represents a set of nodes,*
– *$l_X | X \in \{id, t, d, m, s\}$ is a set of labelling functions, giving an element identifier, its type, its description, its metadata, and its status,*
– *$\rightarrow$ represents the connector.*

We denote by $\{s, g, e, a, j, c\}$ the node types strategy, goal, evidence, assumption, justification, and context respectively. :

– *$l_{id} : N \rightarrow string$ gives the node identifier,*
– *$l_t : N \rightarrow \{ s, g, e, a, j, c\}$ gives the node types,*
– *$l_d : N \rightarrow string$ gives the node descriptions,*
– *$l_m : N \rightarrow P(A)$ gives the node instance attributes,*
– *$l_s : N \rightarrow P(\{tbd\})$ gives node development status.*

In addition, we consider the operation $isRoot(\rightarrow, r)$ which check if the node r is a root.

Our syntactic analysis is based on structural criteria that are checked in two phases: *on-creation criteria* and *post-creation criteria.*

4.1 On-creation criteria

1. Goals cannot be connected directly to other goals in the assurance argument:

$$\forall n, m \in N, (n \rightarrow m) \wedge [l_n = g] \Rightarrow l_t(m) \in \{s, e, a, j, c\}$$

2. Strategies cannot be directly linked to other strategies or evidence in the assurance argument:

$$\forall n, m \in N, (n \rightarrow m) \wedge [l_n = s] \Rightarrow l_t(m) \in \{g, a, j, c\}$$

3. Only goals and strategies are permitted to be undeveloped in the assurance argument:

$$\forall n, m \in N, tbd \in l_s(n) \Rightarrow l_t(n) \in \{g, s\}$$

4. A node cannot establish a connection with itself:

$$\forall n, m \in N, (n \rightarrow m) \Rightarrow l_{id}(n) \neq l_{id}(m)$$

5. Each element has a unique identifier

$$\forall n, m \in N, l_{id}(n) \neq l_{id}(m)$$

4.2 Post-creation criteria

1. Root of the GSN should be a goal:

$$isRoot(\rightarrow, r) \Rightarrow l_t(n) \in \{g\}$$

2. Connectors are only permitted to originate from strategies or goals:

$$\forall n, m \in N, n \rightarrow m \Rightarrow l_n \in \{s, g\}$$

5 Semantic Analysis

The semantic analysis of a GSN may involve interpreting and understanding the meaning of elements present in the GSN model. This process includes identifying relationships between different objectives, justifications, and strategies, as well as semantic interpretation of the context in which they are placed. In this work, we use semantic analysis to verify certain behavioural properties. In that regard, we propose to model the GSN using a formal model i.e., MDP. The choice of MDP is supported by the existence of a probabilistic model-checker that can automatically verify MDP against properties expressed in PCTL. In addition, MDP will allow us to extend the current work in order to evaluate a GSN based on a *confidence score* [10]. We present the main steps of the semantic analysis in Fig. 4.

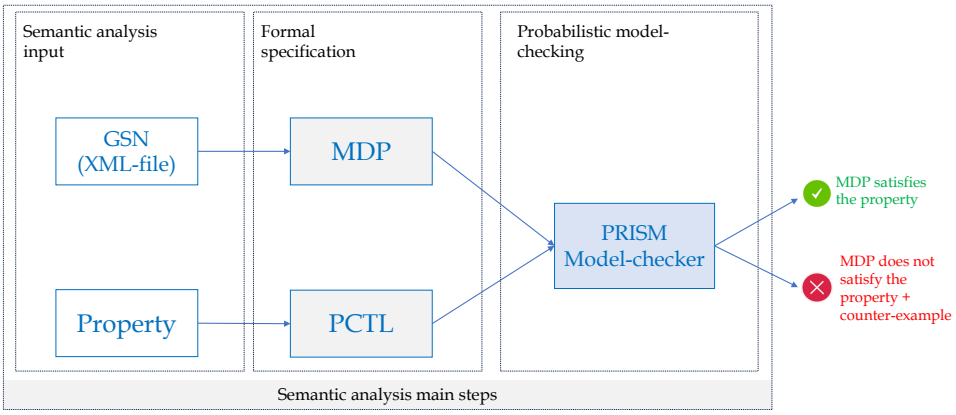

Fig. 4. Semantic analysis main steps.

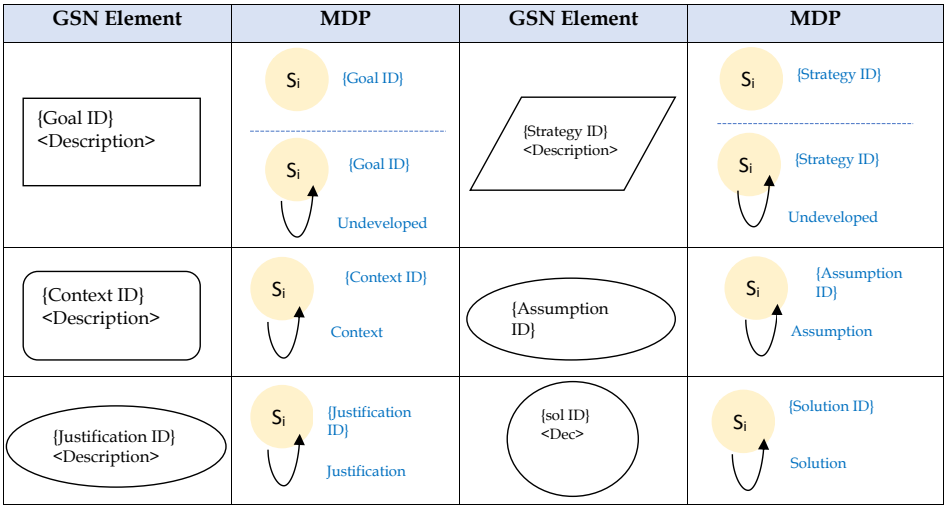

Fig. 5. Mapping of the main GSN elements to MDP states.

5.1 Translate GSN to MDP

Figure 5 summarises the mapping of the main elements of GSN to MDP states. Mainly, we map each GSN element to a state in MDP, and we label this state with the ID. GSN elements *Context, Assumption, Justification*, and *Solution* are represented by an MDP state and loop to this state. For undeveloped *Goal* and *Strategy*, they are represented by a state with a loop labeled "Undeveloped".

After formally specifying the GSN as an MDP, we use PRISM [11] model-checker to verify this MDP against a probabilistic property. SGSN supports properties expressed in PCTL. An example of a property is to verify that all the sub-goals are supported by solutions unless specified as undeveloped.

$$P_{\geq 1}[G(true \cup \text{“}Solution \vee Undeveloped\text{”})] \tag{1}$$

5.2 Running example

In order to illustrate the formal specification of GSN, we consider the GSN illustrated in Figure 2, where Figure 6 represents the MDP formally specifying this GSN. In this

example, we consider the probability associated with each action as 1 since we do not consider the confidence score in this paper. However, for the confidence argument [10], the probabilities could be specified to infer the confidence score.

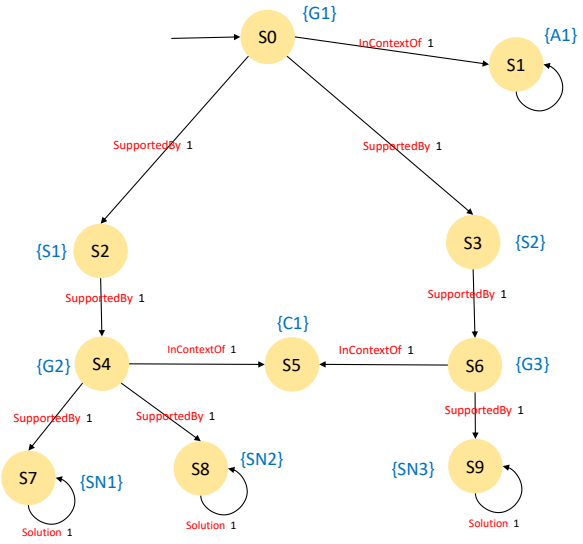

Fig. 6. MDP representing GSN in Fig. 2.

6 Related works

Tools such as [12]–[14] sfacilitate manual or partly automated creation, instantiation, and management of assurance cases. Approaches built upon the AMASS platform [15]–[17] utilize contracts for automated assurance case creation, promoting compositionality and reusability of argumentation patterns. However, these tools do not propose syntactic and semantic analysis beyond automation.

The authors of [18] presented a set of tools supporting the automatic creation of assurance cases, where they showcase this using the tool AdvoCATE [19], however, they did not specify the formalism of the confidence argument.

The Evidential Tool Bus (ETB) [20] offers a framework for tool integration and workflow formalization to oversee the claims and evidence produced by formal tools, aiding assurance activities. The ETB is specifically designed to handle evidence generated in support of tool-specific inferences and for claims where formal tools have been employed, providing a mechanical approach to evidence management. However, this tool does not support the confidence argument, and the formal specification of context, assumption, and strategies.

7 Conclusion

In this paper, we presented Structured GSN (SGSN), an open-source tool to create and edit assurance cases. In addition to basic functionalities related to the creation of GSN, we added two types of analysis, syntactic analysis which checks if the developed GSN satisfies structured criteria, and semantic analysis which checks compartmental properties.

For the semantic analysis, we mapped the GSN to a formal model i.e., Markov Decision Process (MDP), and checked properties expressed in PCTL using PRISM probabilistic model-checker. The formal specification of GSN using MDP allows the extension of the current work to verify a larger set of properties related to the confidence argument.

We aim to enhance the existing version of SGSN to accommodate dynamic updates in the assurance argument. This extension can be particularly advantageous for systems reliant on AI, where claims and evidence may change during runtime based on operational data.

Acknowledgments

We would like to thank our students Alouani Mounir Yacine, and Ouail Mohamed Sedik for their participation in the design and implementation of the SGSN tool.

References

[1] A. Robertson, "European organisation for the safety of air navigation (eurocontrol)," *Annuaire Europeen/European Yearbook: Vol. XXVII*, pp. 618–629, 1981.

[2] R. Palin, D. Ward, I. Habli, and R. Rivett, "Iso 26262 safety cases: Compliance and assurance," 2011.

[3] R. E. Bloomfield, G. Fletcher, H. Khlaaf, L. Hinde, and P. Ryan, "Safety case templates for autonomous systems," *ArXiv*, vol. abs/2102.02625, 2021.

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

[5] R. Bloomfield and P. Bishop, "Safety and assurance cases: Past, present and possible future–an adelard perspective," in *Making Systems Safer: Proceedings of the Eighteenth Safety-Critical Systems Symposium, Bristol, UK, 9-11th February 2010*, Springer, 2009, pp. 51–67.

[6] A. C. W. Group *et al.*, "Goal structuring notation community standard vestion 3," Technical Report SCSC-141BA v2. 0, Safety Critical Systems Club, 2018. `https://scsc.uk/r141C:1?t=1`, Tech. Rep., 2021.

[7] C. C. White III and D. J. White, "Markov decision processes," *European Journal of Operational Research*, vol. 39, no. 1, pp. 1–16, 1989.

[8] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton, "Verifying continuous time markov chains," in *Computer Aided Verification: 8th International Conference, CAV'96 New Brunswick, NJ, USA, July 31–August 3, 1996 Proceedings 8*, Springer, 1996, pp. 269–276.

[9] R Bouchekir, M Guzman, A Cook, J Haindl, and R Woolnough, "Formal verification for safe ai-based flight planning for uavs," in *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, IEEE, 2023, pp. 275–282.

[10] E. Denney, G. Pai, and I. Habli, "Towards measurement of confidence in safety cases," in *2011 International Symposium on Empirical Software Engineering and Measurement*, IEEE, 2011, pp. 380–383.

[11] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, G. Gopalakrishnan and S. Qadeer, Eds., ser. LNCS, vol. 6806, Springer, 2011, pp. 585–591.

[12] M. R. Barry, "Certware: A workbench for safety case production and analysis," in *2011 Aerospace conference*, IEEE, 2011, pp. 1–10.

[13] Y. Matsuno, "D-case editor: A typed assurance case editor," *University of Tokyo*, 2011.

[14] L. Adelard, *Assurance and safety case environment (asce)*, 2011.

[15] I. Sljivo, B. Gallina, J. Carlson, H. Hansson, and S. Puri, "Tool-supported safety-relevant component reuse: From specification to argumentation," in *Reliable Software Technologies–Ada-Europe 2018: 23rd Ada-Europe International Conference on Reliable Software Technologies, Lisbon, Portugal, June 18-22, 2018, Proceedings 23*, Springer, 2018, pp. 19–33.

[16] I. Šljivo, G. J. Uriagereka, S. Puri, and B. Gallina, "Guiding assurance of architectural design patterns for critical applications," *Journal of Systems Architecture*, vol. 110, p. 101 765, 2020.

[17] J. L. de la Vara, A. Ruiz, and G. Blondelle, "Assurance and certification of cyber–physical systems: The amass open source ecosystem," *Journal of systems and software*, vol. 171, p. 110 812, 2021.

[18] E. Denney and G. Pai, "Tool support for assurance case development," *Automated Software Engineering*, vol. 25, no. 3, pp. 435–499, 2018.

[19] E. Denney, G. Pai, and J. Pohl, "Advocate: An assurance case automation toolset," in *Computer Safety, Reliability, and Security: SAFECOMP 2012 Workshops: Sassur, ASCoMS, DESEC4LCCI, ERCIM/EWICS, IWDE, Magdeburg, Germany, September 25-28, 2012. Proceedings 31*, Springer, 2012, pp. 8–21.

[20] S. Cruanes, G. Hamon, S. Owre, and N. Shankar, "Tool integration with the evidential tool bus," in *International Workshop on Verification, Model Checking, and Abstract Interpretation*, Springer, 2013, pp. 275–294.