Assurance Cases Across the Product Line

Sara Magdalinski
Critical Systems Labs Inc.
Vancouver, Canada
sara.magdalinski@cslabs.com

Annie Chung
Critical Systems Labs Inc.
Vancouver, Canada
annie.chung@cslabs.com

Laure Millet
Critical Systems Labs Inc.
Vancouver, Canada
laure.millet@cslabs.com

***Abstract*— Modern automotive product lines evolve systematically: a common baseline provides the foundation for increasing levels of functionality and performance, while variability points enable products to address different requirements and configurations. The same principle can be extended to safety assurance. By systematically managing variability, the efficiencies of a shared product-line baseline can carry over into the assurance case. This paper presents an approach for managing variability within a common assurance model, in which invariant and variant assurance information is maintained together. Variability points represent differences in applicability, while projections provide focused views for individual product variants. Using an example of a hypothetical automated-driving perception system product line, this paper illustrates how a single assurance model can be maintained across multiple product variants while accommodating differences in argumentation, evidence, and monitoring information.**

Keywords*— *assurance case, goal structuring notation (GSN), product-line engineering, safety assurance, variability management

I. Introduction

Product lines are increasingly common in domains such as automotive systems, where related products share a substantial design baseline while differing in functionality, performance, and operating conditions. Product-line engineering provides a systematic way to manage this commonality and variability. In automotive advanced driver-assistance systems (ADAS), for example, variability has been represented across operational, functional, and constructional perspectives [1]. When these differences affect safety, they can also affect the assurance information required for individual product variants.

The same idea of managing commonality and variability can be applied to safety assurance. Two product variants can share much of their architecture and safety rationale, but differences in functionality, hardware and software components, operational design domain (ODD), allocation of responsibility, or fallback behaviour can affect what needs to be demonstrated for each product. For example, an automated-driving function that relies on a human driver to resume control requires claims and evidence relating to notification, takeover time, and transition management. A variant that cannot rely on a driver, by contrast, must demonstrate that the system can independently reach a minimum-risk condition. Other safety goals might remain unchanged across variants while depending on different analyses, tests, assumptions, or operational monitoring.

One option is to maintain a separate assurance case for each product variant. This becomes inefficient, however, when the variants share a large part of the same argument. Claims and evidence can be duplicated, changes to shared information must be reflected in several cases, and independently maintained arguments can gradually become inconsistent.

An alternative is to maintain a common assurance argument that separates elements shared across the product line from those that apply only to particular variants. This begins by identifying the safety goals for the product line, determining which goals are common and which vary, and defining their applicability to individual variants. Doing so makes the relationship between product variability and assurance variability explicit, rather than allowing these differences to emerge independently in separate safety cases.

Variability points can then be associated with elements of the common assurance model whose applicability changes across the product line. For each variability point, variables representing the product variants are assigned values indicating whether the corresponding element applies to a given variant. Although safety goals provide a natural starting point for identifying variability, other parts of the argument can also differ. Solution nodes, for instance, can refer to different evidence for different variants, while supporting artifacts and safety performance indicators can also have variant-specific applicability. Variability is therefore captured in the underlying assurance model rather than being limited to the graphical structure of the argument.

This paper examines an approach for managing such variability within a common assurance model. Once the variability points and their associated assignments have been defined, projections can be used to produce variant-specific views of the common argument. Each projection shows the invariant elements together with the variant elements that apply to the selected product variant. Using a hypothetical product line for an automated-driving perception system, the paper demonstrates how invariant and variant safety goals can be identified, how variability points can be applied to goals, solutions, artifacts, and indicators, and how projections can provide focused assurance views for individual product variants.

II. GSN-Style Safety Case Argumentation

Safety case arguments can be presented as written prose—that is, as sentences organized into paragraphs. However, written prose does not easily scale to the complexity of ADAS and automated driving systems. Its limitations would be even more apparent in a safety case for a product line, where some details might not be relevant to a given variant.

An alternative to written prose is a graphical notation such as Goal Structuring Notation (GSN) [2] or Eliminative

Argumentation (EA) [3]. Using a graphical notation does not necessarily exclude written prose. In some cases, the graphical representation of a safety case argument might supplement the written prose; in others, the reverse might be true. For example, written prose might supplement the graphical representation by summarizing the argument and highlighting only its subtleties.

In simple terms, GSN consists of a set of different kinds of "building blocks" that can be assembled into a structure in which a top-level goal is decomposed through multiple levels into a set of bottom-level sub-goals. To complete the argument, these bottom-level sub-goals are linked to solutions, such as references to test results in a test report.

To make effective use of GSN, it helps to have a general strategy for developing the overall structure of the argument. A variety of such strategies exist, including public facing approaches such as the MISRA Guidelines for Automotive Safety Arguments (GASA) [4].

For the example described later in this paper, the core strategy used for the high-level structure of the GSN-style argument can be summarized as follows:

1) The argument is decomposed into different "lines of argumentation" corresponding to general categories of risk. For example, one such category is cybersecurity risk.

2) Next, the argument is further decomposed to account for previously identified goals, including invariant goals that apply across the product line and variant goals that apply only to certain variants.

3) Finally, the argument is completed through additional decomposition to the level of sub-goals that can be directly addressed by evidence.

The strategy above results in the creation of a common argument that extends across the product line. Going beyond the conventional use of GSN to represent safety arguments, a final step involves associating variant elements of the argument with variables. Assigning values to these variables determines which parts of the common argument apply to a given product-line variant.

III. Managing Variability in Product-Line Assurance

Managing assurance across a product line requires a way to preserve the information that is shared between products while representing the differences that affect individual variants. The approach described here maintains this commonality within a single assurance model and introduces variability only where the applicability of assurance information differs.

A. Identifying Invariant and Variant Safety Goals

Managing variability in an assurance case begins with the safety goals established for the product line and determining how those goals apply to its individual variants. Let the set of valid product variants be $V = \{v_1, v_2, \dots, v_n\}$ and the set of identified safety goals be $G = \{g_1, g_2, \dots, g_m\}$.

The safety goals are identified through the normal safety engineering process, drawing on hazard analysis, risk assessment, safety requirements, system architecture, and operational constraints. Product-line assurance builds on these activities by considering how the resulting safety obligations apply across the different members of the product line.

A difference in a product line becomes relevant to the assurance model when it changes what must be claimed, assumed, argued, demonstrated, or monitored to establish safety. The first step is therefore to determine which safety goals apply to each product variant.

For a safety goal g and product variant v, applicability can be represented as:

$$A(g, v) \in \{required, not\ required\}. \quad (1)$$

Here, $A(g, v)$ indicates whether goal g applies to variant v. Once this has been determined for each variant, the set of variants to which a particular goal applies can be defined as:

$$R(g) = \{v \in V | A(g, v) = required\}. \quad (2)$$

$R(g)$ therefore represents the product variants that require goal g. If $R(g) = V$, the goal applies across the entire product line and is treated as invariant. If $R(g)$ contains only a subset of V, the goal is variant.

The objective is to preserve as much of the common assurance model as possible. If a goal applies to all variants, it remains invariant, and the analysis continues into the supporting argument until a point of divergence is reached. Variability is introduced at that level, rather than making an entire branch variant-specific because of a lower-level difference. This preserves the common structure while making differences between variants explicit.

B. Variability in the Supporting Argument

After the applicability of the identified safety goals has been established, the supporting argument for each invariant goal can be examined. A goal can apply across the entire product line even if the reasoning used to support it differs between variants.

In a GSN-style argument, these differences can appear in strategies, lower-level goals, contexts, assumptions, or solutions. Each supporting element is considered in relation to the variants for which its parent claim applies. If it applies to all those variants, it remains invariant; if its applicability differs, it is treated as variant.

The analysis then continues through the argument in the same way, preserving common elements until a difference is reached. Variability can also occur in supporting assurance information without changing the argument structure itself. For instance, a common solution might rely on different evidence for different variants, or a common claim might be associated with different monitoring data sets collected during operational use. These differences can be represented separately within the assurance model.

C. Representing Variability with Variability Points

Differences in applicability can be represented using variability points. A variability point is associated with an assurance element and contains a set of variability-point

assignments (VPAs) that define how that element relates to the variables used to represent product-line variability.

Let $X = \{x_1, x_2, \dots, x_k\}$ denote these variables. A variable can represent either a complete product variant or an individual feature, depending on how the product line is modelled. For a predefined set of products, each variant can have its own variable. Alternatively, variables can represent individual features, with a product variant described by the combination of features that apply to it.

For a variability point p and variable x, an individual variability-point assignment is represented as:

$$VPA(p, x) \in \{True, False\}. \quad (3)$$

Let $X_p \subseteq X$ denote the variables assigned at variability point p. The variability point can then be represented as the set:

$$VP(p) = \{(\mathrm{x}, VPA(p, x)) \mid x \in X_p\}. \quad (4)$$

The assignments within a variability point are considered together. For product-level variables, the assignments indicate which variants the element applies to; for feature-level variables, they define the feature conditions under which it applies.

Elements that apply across the relevant product line remain invariant and do not require a variability point. This allows the common assurance model to retain its shared structure while explicitly representing where particular elements depend on product variants or features.

D. Variability Beyond the Graphical Argument

Variability need not be limited to producing different versions of the same GSN argument. A GSN diagram provides a graphical representation of an assurance argument, while additional supporting information might exist beyond the visible nodes and relationships.

Differences between product variants might also affect supporting assurance information, such as test results, analyses, verification records, performance data, or evaluation thresholds, without changing the structure of the argument. Representing variability within the underlying assurance model allows these differences to be captured directly while retaining the parts of the argument that remain applicable across the product line.

Artifacts and indicators are two examples where this is particularly relevant. Artifacts can represent evidence or other engineering information associated with the argument. Indicators can connect key performance indicator (KPI) or safety performance indicator (SPI) data to argument nodes and evaluate that data against defined thresholds. Their applicability might differ across variants or features even when the associated GSN nodes remain unchanged.

E. Variability in Supporting Assurance Information

Where supporting information differs across variants or features, variability can be applied directly to that information rather than to the surrounding argument structure.

For artifacts, this can allow the evidence associated with a claim or solution to vary independently of the node itself. A common solution might remain applicable across the product line while test results, analyses, or verification records apply only to certain variants or features. Associating variability with the artifact can capture this distinction without requiring the shared argument to be duplicated.

The same principle can apply to indicators. A common claim can be associated with different indicators depending on the variant or feature being considered. An indicator can contain the data and threshold rules used to evaluate the relevant performance measure. Treating the indicator as a point of variability can therefore allow the appropriate monitoring information to be associated with the conditions under which that indicator is relevant.

Artifacts and indicators are particularly relevant because differences in their applicability can change the substantive assurance information available for a variant or feature. Artifacts can provide evidence used to support the argument, while indicators can provide information used to assess or monitor safety performance. Other model elements, such as labels or terminology, can support organization and interpretation without directly changing the evidence or monitoring associated with a particular variant. Artifacts and indicators therefore demonstrate why variability is more effectively represented at the assurance-model level rather than being limited to the graphical argument alone. This allows variability to be associated with the supporting information that differs, while preserving the common argument structure where appropriate.

IV. Projections

Representing variability within a common assurance model makes it possible to automatically derive focused views for specific products or feature combinations. These projections are generated from the common model using the selected variable assignments, so that the applicable assurance information is presented without maintaining separate models.

A. Definition and Purpose of Projections

Projections provide focused views of the common assurance model by combining invariant elements with the variant elements that apply under a selected set of variable assignments. A projection is derived by evaluating variability points against those assignments, rather than being created as a separate assurance argument. The common assurance model is maintained as the source, while each projection presents the model elements applicable under the selected variable assignments

For product-line assurance, this allows both invariant and variant elements to be maintained within a single common assurance model. Variability points capture where applicability differs, and projections use that information to present the parts of the model relevant to a particular product variant or feature combination.

B. Deriving Projections from the Common Argument

A projection is defined by assigning Boolean values to selected variables in the common assurance model. Recall that $X = \{x1, x2, \dots, xk\}$ denotes the set of variables used to represent product-line variability. Let $X\sigma \subseteq X$ denote the variables assigned in a particular projection, and let $\sigma(x)$ denote

the value assigned to variable x. A projection does not need to assign a value to every variable in X.

When a projection is generated, its assignments are compared with the variability-point assignments associated with each variability point. An element associated with variability point p is included if the projection assignments do not conflict with those assignments. Formally:

$$\forall x \in X_\sigma \cap X_p, \sigma(x) = VPA(p, x) \quad (5)$$

In other words, where a variable is assigned in both the projection and the variability point, the values must agree. A mismatch excludes the associated element, while variables left unassigned in the projection do not affect the evaluation.

For example, suppose a variability point p assigns $a = True$ and $b = True$. A projection assigning $a = True$ and $b = False$ excludes the associated element because the value of b conflicts. In contrast, a projection assigning only $a = True$ includes the element because no conflicting value for b is specified.

Invariant elements are not subject to this comparison and remain part of the projection. Conceptually:

$$\mathrm{P}(\sigma) = \mathrm{E}_{\mathrm{inv}} \cup \mathrm{E}_{\mathrm{v}}(\sigma) \quad (6)$$

where E_{inv} represents the invariant elements of the common model and $E_v(\sigma)$ represents the variant elements whose variability-point assignments are compatible with σ.

This evaluation applies to all elements, including argument nodes, artifacts, and indicators that have variability points. A projection can therefore reflect variability in both the argument structure and its supporting assurance information.

C. Product- and Feature-Level Projections

The meaning of a projection depends on how the variables have been defined. For a product line consisting of a known set of product variants, each variant can be represented by a variable, and a projection can provide the assurance view for that particular member of the product line. This is the approach used later in the Ilgaak example.

Variability can also be represented at the feature level. If individual features are represented as variables, a projection can be derived for a combination of those features. This avoids requiring every possible product to be represented explicitly as a separate variable and allows the assurance model to follow the way variability is structured within the product line.

In either case, the projection combines the invariant elements with the variant elements applicable to the selected product or feature combination. The difference lies in the level at which the underlying variables describe the product line.

D. Functional and Operational Variability

Projections can reflect different dimensions of variability within a product line. Differences in functionality, for example, might introduce or remove safety obligations and can change the supporting argument or evidence required for a product. Where these differences are represented through variables and variability points, a projection can present the assurance information associated with the relevant functionality.

Operational differences can be represented in the same way. Product variants can operate within different operational design domains (ODDs), and these differences can affect the claims, assumptions, evidence, or monitoring required to establish safety. A projection can therefore reflect the assurance information applicable to a particular set of operational conditions.

Functionality and ODD are two examples of characteristics that can influence the contents of a projection. Other product features or characteristics can be represented through the same mechanism where they affect the applicability of assurance information.

E. Projections for Stakeholder Access

The same projection mechanism can also support views that are not based on differences between members of a product line. One example is controlling the information presented to different stakeholders.

Suppliers, customers, assessors, and certification authorities can require different levels of access to the information contained within an assurance case. A projection can provide a focused or restricted view of the common model according to those information-sharing requirements.

For instance, the supplier of a product line might extend the assurance case to incorporate some confidential information about Intellectual Property (IP) that has been shared by a customer. Variability can be used to restrict the visibility of this confidential information to a projection that can only be accessed by this particular customer. The same approach could be used for confidential information shared by other customers.

This use is distinct from a product- or feature-level projection. A product-line projection reflects differences in which assurance information is applicable to the product being considered, whereas an access-oriented projection reflects differences in which information is presented to a particular stakeholder. Both can be derived from the same underlying assurance model, but they serve different purposes.

V. Application to the Ilgaak Product Line

This section illustrates how the proposed approach can be applied in practice. Using the Ilgaak product line, it follows the progression from structuring a common assurance argument and identifying variability to applying variability points and deriving projections for individual product variants.

A. Ilgaak Product Line and Assurance Model

To illustrate the approach described in the preceding sections, a hypothetical automated-driving perception system product line, Ilgaak, was developed. Ilgaak consists of four product variants: Assist, Drive, Pilot, and Pilotless. The variants share a common baseline while differing in functionality, sensor capabilities, operational design domains (ODDs), and approaches to failure response. These differences provide a useful example of how invariant and variant assurance information can be represented within a common assurance model.

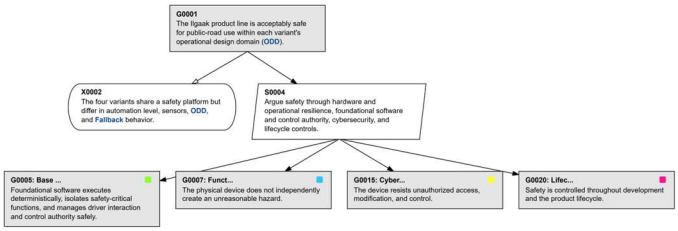


Fig. 1. Top-level structure of the common Ilgaak assurance argument in Socrates.

The Ilgaak assurance model was developed using *Socrates – Assurance Case Editor* [1] product, an assurance-case development and maintenance tool that supports structured assurance models, including GSN-style argumentation. For this example, Socrates is used to maintain a single common assurance model containing both invariant and variant elements, together with associated artifacts and indicators. Variability points and variables represent differences in applicability across the product line, and projections are derived by combining the invariant elements with the variant elements applicable to each product variant.

The example is intended to demonstrate the management of assurance variability rather than provide a complete safety analysis of the four products. The figures in the following sections are Socrates screenshots. They show how the concepts introduced earlier in the paper are represented in the Ilgaak model.

B. *Structure of the Common Argument*

The Ilgaak model begins with a common top-level claim that the product line is acceptably safe for public-road use within the ODD applicable to each variant. The argument is organized into four principal lines of argumentation: Base Software and OS Safety, Cyber Security Resilience, Functional Integrity, and Lifecycle and Process Quality.

These lines provide a common structure for the product line rather than a separate argument structure for each variant. Elements that apply throughout the product line form the invariant part of the argument, while differences are introduced only where the applicability of a goal or its supporting information changes between variants.

Figure 1 shows the high-level Ilgaak argument in Socrates. At this level, the argument provides a shared structure from which the applicability of more detailed goals and supporting elements can be considered.

C. *Identifying Invariant and Variant Goals*

Following the method described in Section III, goals in the Ilgaak argument are considered according to their applicability across the four product variants. Goals that apply across the product line remain invariant, while goals whose applicability differs are treated as variant.

This is illustrated in Figure 2 within the Base Software and OS Safety line of argumentation. The higher-level strategy and several supporting goals remain common, while variability is introduced further down the argument where the safety responsibilities of the individual products differ. For example, Driver Takeover applies to Drive and Pilot, while Reliable and Timely Driver Alerts applies to Assist, Drive, and Pilot. The pink-coloured badges that contain a Y-shaped "fork in the road" symbol indicate nodes in Figure 2 with associated variability points.

This illustrates how an invariant higher-level argument can be retained while variability is introduced only where applicability changes. The common structure is therefore preserved rather than making an entire line of argumentation variant-specific.

D. *Applying Variability Points*

The identified differences are encoded in the Ilgaak model using variability points. In this example, Assist, Drive, Pilot, and Pilotless are represented as variables, with Boolean assignments at each variability point indicating the product variants associated with that element.

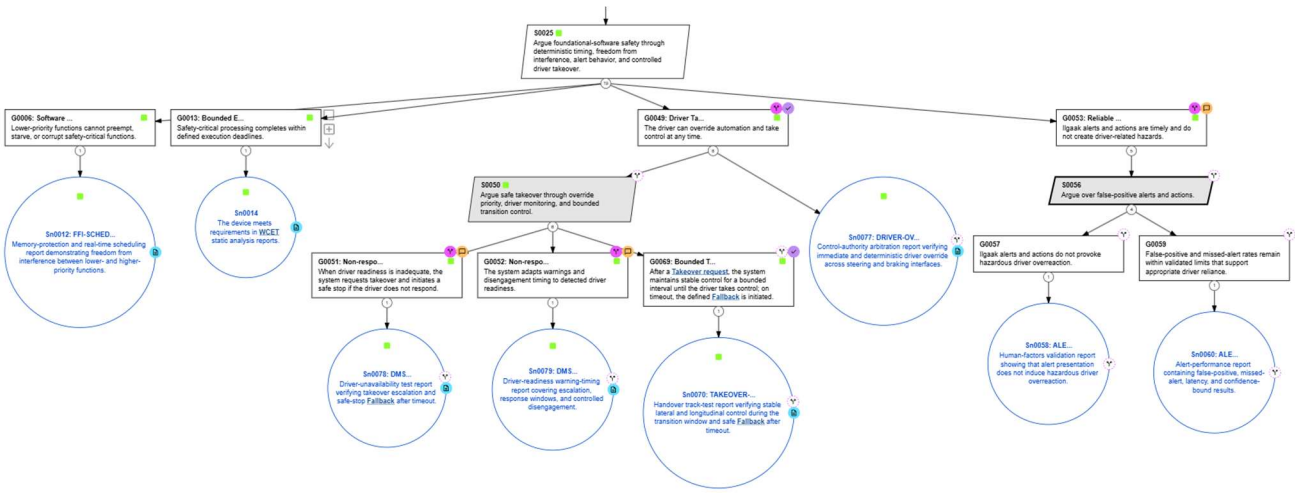


Fig. 2. Invariant and variant elements within the Base Software and OS Safety line of argumentation. Pink symbols identify nodes with variability points.

[1] https://criticalsystemslabs.com/socrates-assurance/

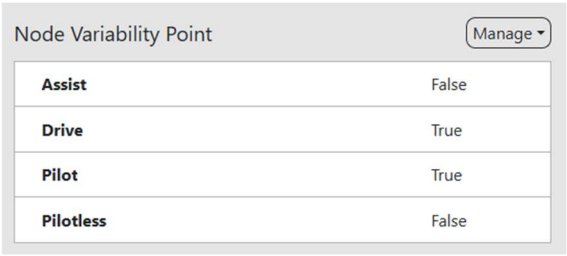


Fig. 3. Variability point for the Driver Takeover goal in Socrates.

Figure 3 shows the variability point for the Driver Takeover goal. Consistent with the applicability identified above, Drive and Pilot are assigned True, while Assist and Pilotless are assigned False. The variability point records an applicability decision established through the preceding safety and assurance analysis. Invariant elements do not require variability points because their applicability does not differ across the product line. Although Ilgaak uses variables representing complete product variants, the same mechanism can also be used with feature-level variables, as described in Section III. The mechanism can also be applied to supporting assurance information such as artifacts and indicators, which is discussed in the following subsection.

E. Variability in Supporting Assurance Information

Variability in the Ilgaak model is not limited to GSN nodes. It may also be applied to artifacts and indicators, allowing supporting assurance information to vary without changing the surrounding argument structure.

An ODD Exit goal provides an example. The goal remains common across the product line, but the evidence used to support it differs according to the response required for each product variant. The common assurance model contains four variant-specific artifacts addressing the applicable ODD-exit response for Assist, Drive, Pilot, and Pilotless. Variability is associated directly with these artifacts so that, when a projection is generated, only the artifact applicable to the selected product variant is included. The common goal and argument structure are therefore retained across the product line, while each projection presents the supporting evidence relevant to that variant.

Indicators can be handled in the same way. The Ilgaak model includes indicators associated with ODD-boundary behaviour, such as boundary-detection results. An indicator can contain the data and threshold rules used to evaluate a performance measure, while its variability point identifies the product variants for which that indicator applies. This illustrates why variability is represented at the assurance-model level rather than only in the graphical argument. Differences in evidence and performance assessment can be captured where they occur without unnecessarily making the associated GSN nodes variant-specific.

F. Ilgaak Projections

The Ilgaak model defines four projections corresponding to Assist, Drive, Pilot, and Pilotless. Each is derived from the same common assurance model using the variable assignments defined for that product variant.

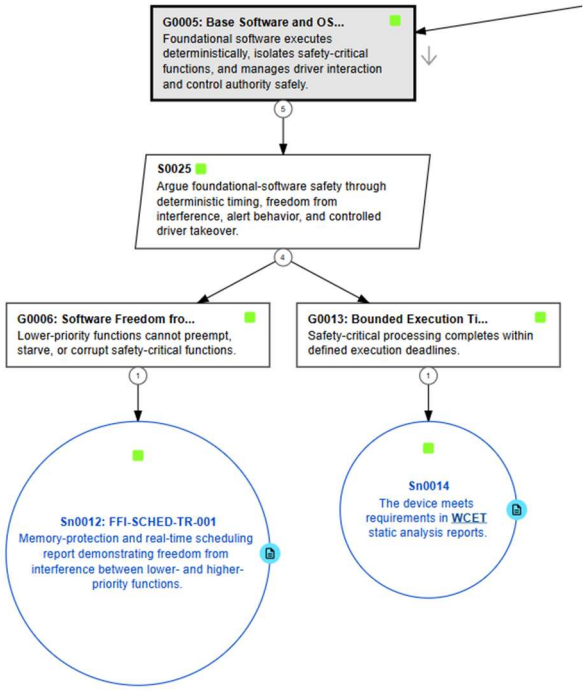


Fig. 4. Assist projection of the Base Software and OS Safety line of argumentation.

Figure 4 shows the Assist projection of the Base Software and OS Safety line of argumentation introduced earlier. Invariant elements remain in the projection, while variant elements are included only when their variability-point assignments are compatible with the Assist projection. For example, Driver Takeover is not included because it applies to Drive and Pilot rather than Assist. Supporting artifacts and indicators with variability points are evaluated in the same way.

The projection therefore provides a focused view of the assurance information relevant to Assist without creating or maintaining a separate assurance argument. The same common model is used to derive the corresponding views for Drive, Pilot, and Pilotless.

VI. Related Work

Prior work on product-line assurance has considered how common and variable assurance information can be represented and reused across related system variants. Approaches differ in how variability is related to the underlying system and how product-specific assurance arguments are obtained.

In [5], GSN patterns and modular safety arguments were used to support the assurance of configurable architectures in safety-critical product lines. Variability in the safety argument is related to variability in the underlying architecture, allowing common argumentation to be separated from argumentation associated with particular product variants. In [6], this idea was extended toward the automated generation of modular product-line safety cases, using traceability between variability in the product design and safety analysis and the

resulting safety-case structure. Contract-based design was used in [7] together with modular GSN patterns to construct assurance cases at the product-line level and support reuse across related system variants.

In [8], the authors sought to lift the activity of creating assurance cases from system models, such that a single variability-aware assurance case (representing a product line of assurance cases) can be derived directly from a product line model. The authors formalized a model of (product) assurance case development based on formal argument templates and proved that such templates could be rigorously lifted. They also proposed to associate particular argument templates with lifted software analyses, which allows for assurance-relevant variation points to be automatically detected in system models and reflected in the assurance case. This work was later extended in [9], where lifted assurance case templates were exploited to support variability-aware impact analysis, allowing developers to identify sets of products for which particular claims, arguments, and evidence were reusable or required revision following a change to the product line.

The approach presented in this paper shares the objective of managing assurance across a product line without maintaining an independent argument for every product variant. The work in [8] and [9] develops a formal, variability-aware approach to assurance-case construction and evolution. This paper complements that work by demonstrating how product-line variability can be represented directly within a maintained assurance model using variables, variability points, and projections. The approach also demonstrates how the same variability mechanism can be applied beyond the argument structure to supporting artifacts and indicators, allowing differences in evidence and performance assessment to be represented while retaining common argument elements where appropriate.

VII. Conclusion

Product-line development creates a corresponding challenge for safety assurance: related products can share substantial portions of their safety rationale while differing in functionality, operational conditions, system characteristics, or supporting assurance information. Maintaining a separate assurance case for every product variant can obscure this commonality and increase the effort required to keep shared information consistent.

This paper has described an approach in which invariant and variant elements are maintained within a common assurance model. Safety analysis establishes where assurance obligations differ across the product line, and variability points are then used to represent those differences explicitly. Variables can represent complete product variants or individual features, while projections provide focused views by combining the invariant elements with the variant elements applicable under the selected variable assignments.

Variability can also extend beyond the GSN argument structure to supporting artifacts and indicators. Representing these differences at the assurance-model level allows evidence and monitoring information to vary where necessary without unnecessarily varying the surrounding argument. The Ilgaak example, implemented in Socrates, illustrates how this approach can be applied across multiple product variants and used to derive corresponding projections from a single maintained model.

The approach provides a basis for preserving common assurance information while making product-line differences explicit and traceable. It might also support broader applications in which assurance information varies across product versions, operational environments, deployments, or other dimensions, while retaining a shared underlying argument.

Acknowledgements

In addition to the authors of this paper, Simon Diemert, Jeff Joyce, and Torin Viger of Critical Systems Labs Inc. have made fundamental contributions to the concept of variability for assurance case arguments and its implementation in the Socrates tool. Support for this work from the National Research Council of Canada is gratefully acknowledged.

References

[1] J. Lameh, A. Dubray, and M. Jankovic, "Modeling variability in product line engineering (PLE) for systems engineering (SE)," *Proceedings of the Design Society*, vol. 5, pp. 2491–2500, 2025, doi: 10.1017/pds.2025.10263.

[2] T. Kelly and R. Weaver, "The Goal Structuring Notation: A safety argument notation," in *Proc. Workshop on Assurance Cases, Int. Conf. Dependable Systems and Networks (DSN)*, 2004.

[3] J. B. Goodenough, C. B. Weinstock, and A. Z. Klein, *Eliminative Argumentation: A Basis for Arguing Confidence in System Properties*, CMU/SEI-2015-TR-005. Pittsburgh, PA, USA: Software Engineering Institute, Carnegie Mellon University, 2015.

[4] MISRA, *Guidelines for Automotive Safety Arguments*. Nuneaton, U.K.: Motor Industry Software Reliability Association, Sep. 2019.

[5] I. Habli and T. Kelly, "A safety case approach to assuring configurable architectures of safety-critical product lines," in *Architecting Critical Systems*, Lecture Notes in Computer Science, vol. 6150. Berlin, Germany: Springer, 2010, pp. 142–160, doi: 10.1007/978-3-642-13556-9_9.

[6] A. L. de Oliveira, R. T. V. Braga, P. C. Masiero, Y. Papadopoulos, I. Habli, and T. Kelly, "Supporting the automated generation of modular product line safety cases," in *Theory and Engineering of Complex Systems and Dependability*, Advances in Intelligent Systems and Computing, vol. 365. Cham, Switzerland: Springer, 2015, pp. 319–330, doi: 10.1007/978-3-319-19216-1_30.

[7] D. Nešić, M. Nyberg, and B. Gallina, "Product-line assurance cases from contract-based design," *Journal of Systems and Software*, vol. 176, Art. no. 110922, 2021, doi: 10.1016/j.jss.2021.110922.

[8] L. Murphy, T. Viger, A. Di Sandro, and M. Chechik, "PLACIDUS: Engineering product lines of rigorous assurance cases," in *Integrated Formal Methods*, Lecture Notes in Computer Science, vol. 15234. Cham, Switzerland: Springer, 2025, pp. 87–108, doi: 10.1007/978-3-031-76554-4_6.

[9] L. Murphy, T. Viger, A. Di Sandro, A. A. Babikian, and M. Chechik, "Assurance case development for evolving software product lines: A formal approach," *Formal Aspects of Computing*, vol. 38, no. 2, 2026, doi: 10.1145/3796233.