Assurance via Workflow\textsuperscript{+} Modelling and Conformance

Abstract—We propose considering assurance as a model management enterprise: saying that a system is safe amounts to specifying three workflows modelling how the safety engineering process is defined and executed, and checking their conformance. These workflows are based on precise data modelling as in functional block diagrams, but their special distinctive feature is the presence of relationships between the output data of a process and its input data; hence, the name “Workflow Plus”.

A typical WF\textsuperscript{+}-model comprises three layers: (i) process and control flow, (ii) dataflow (with input-output relationships), and (iii) argument flow or constraint derivation. Precise dataflow modelling signifies a crucial distinction of WF\textsuperscript{+}-based and GSN-based assurance, in which the data layer is mainly implicit. We provide a detailed comparative analysis of the two formalisms and conclude that GSN does not fulfil its promises.

Index Terms—Assurance, Safety case, Workflow modelling and conformance, Metamodelling

I. INTRODUCTION

Safety cases (SC) are an established approach to certifying safety-critical systems in many domains [1]. However, the size and complexity, and hence the cost, of a typical SC for a modern computer-controlled system can be enormous and, in some cases, greater than the cost of building the system [2]. This explains the long-standing demand for tooling to assist and automate the building and maintenance of SCs [3–5]. The activity in this domain is mainly based on GSN [6] or CAE [7] diagrams (further collectively referred to as GSN-style diagrams), which has become a de facto standard for documenting SCs. An OMG initiative in this direction, the Structured Assurance Case Metamodel (SACM) [8], can also be seen as an effort to standardize GSN-like notation in MOF/UML terms.

In this paper, we report on several shortcomings of the GSN formalism for building and maintaining SCs. We will show that the shortcomings are the result of two main GSN’s limitations: a) GSN conducts logical inference that (we will show) essentially involves graph-based object-oriented (GBOO) data, in a significantly data-simplified (and at some points, data-ignorant) way. b) GSN manages the basic evidence layer of the argument flow, which (we show) is also a GBOO data enterprise, in an unstructured discrete setting leaving an array of important relationships implicit. We demonstrate with an example (taken from the GSN Community Standard [2]) that these drawbacks can essentially distort the argument flow and thus create a dangerous false sense of confidence. A significant investment of time and money into the development of tools for GSN-style SCs may be a misguided endeavour as GSN-based tools will suffer from the drawbacks of GSN. We see the need for an assurance framework alternative to GSN, which will better achieve GSN’s original focus on structure, decomposition, and argument.

We thus return to first principles by specifying a set of requirements for a foundational framework, on which tooling should be based. There are two obvious major requirements necessary to reduce the cost of building and maintaining SCs.

R1) Provide a careful reusability analysis with multiple reusability layers identified.
R2) Allow for an incremental approach based on extensive and effective traceability mechanisms.

These tasks are typical for MDE, and known to be difficult, but in the assurance context they become especially challenging due to the following challenges [10]:

C1) the heterogeneity of the artifacts to be integrated in the safety argument flow;
C2) with the need to include informal and domain-specific artifacts, and a multitude of implicit assumptions about system behaviour and interaction;
C3) the necessity to coordinate the functional system development and its safety engineering.

Addressing these challenges gives rise to the following additional requirements.

R3) Provide a unified approach to heterogeneous artifacts and argument flow.
R4) Be easily understandable by the stakeholders to allow for effective communication, and be easily extensible to accommodate resulting changes from such communication.
R5) Be system-engineering friendly.
R6) Be safety-engineering friendly, including the necessity to observe reasonable conservatism.

Requirements R3-5 address challenges C1-3 resp. (and R5 actually spans across all the challenge range); we added R6 as especially important in the domain of safety-critical systems, where a reasonable conservatism is a forced necessity [11].

In this paper, we propose a modelling framework referred to as Workflow\textsuperscript{+} (briefly, WF\textsuperscript{+}) which meets the requirements on the specification level and shows promising potential in ongoing collaboration with our automotive partners. Tool implementation has not started yet. The goal of the paper is to lay theoretical foundations for applying established MDE methods to the assurance of safety-critical software intensive systems. We believe that this can lead to improved rigour, conceptual and technological clarity, and advanced tool support for both development and assurance of such systems.
WF+ employs the idea of a process as a unifying foundation for modelling and analysis of both cyber-physical systems and SEPs (safety engineering processes) that include verification and validation (V&V) of the former analysis. We refer to the framework as WF+ due to a special feature of a typical SEP subprocess: its output data are related to its input data rather than disjoint from them as in the classical block diagram setting. The name WF+ refers to both a workflow composed from processes with input-output related data, and the approach to SC development based on such workflows.

In the following, we describe the contents of the paper and list its technical contributions. In Sect. II, we give a formalized definition of assurance based on the notions of WF+-conformance and refinement. We are not aware of such model-based assurance definitions in the literature. In Sect. III-IV, we show how the WF+-method works with an example: we take a GSN diagram from the GSN Community Standard [9], interpret it as an SC, recover its semantics, rearrange it in WF+-terms, and rebuild the case as WF+-conformance. We show that the argument flow—a focal point of the GSN-based assurance—is a data-driven enterprise. This significantly differentiates WF+ from GSN, in which data and dataflow are mainly implicit. Sect. V is a comparative analysis of WF+ vs. GSN: it can be seen as an evaluation of the new framework against an established one based on the standard example. We show that the argument flow in the example is significantly distorted—a disappointing observation for an example residing in a popular standard since 2011, Version 1 [6, p.5] until now in Version 2 [9, p.14]. In Sect. VII, we discuss how well WF+ addresses the issues described in the empirical survey [12] and satisfy our requirements above. Sections VI-VIII are brief overviews of our work in progress and future work, and related work resp. Sect. IX concludes.

We extensively use colours in our diagrams as a visually convenient additional way to convey semantics of diagram elements, which is encoded by labels (specifically, UML’s stereotypes) and shapes of elements.

II. ASSURANCE AS CONFORMANCE

In this section we introduce our general model of assurance.

A. BACKGROUND

Approaches to certification of safety-critical software systems has traditionally been divided into two (somewhat overlapping) groups: process-based and product-focused (cf. [11]). An obvious plus of the former is the clarity and practicality of the verification: the regulator can always check whether everything prescribed was really executed. An obvious minus is indirectness: if even everything prescribed was executed, it is unclear how much it guarantees that the product as such is safe. The situation is reversed for product-focused assurance, which assumes that safety can be specified as a collection of attributes such that having their measurable values within a prescribed range can be interpreted as having sufficient evidence that the system is safe. If such a collection could be found, it would provide quite a direct approach to assurance but, unfortunately, reducing safety to an a priori defined finite collection of measurable attributes is unrealistic for complex systems (unless magic creatures like Laplace’s daemon are invoked). Indeed, the qualitative and quantitative infinity of possible interactions of such a system as a modern car with its environment (other cars, pedestrians, weather and road conditions, and the driver) makes any hazard analysis performed with a finite number of models and simulations within a finite time period potentially incomplete.

Moreover, any attempt to mitigate this problem with super-intelligent mathematical models amenable to super-powerful analysis techniques is doomed to fail as even a perfect “proof” of a system’s safety is necessarily carried out within some reasoning system R (be it a logic, a math model, a set of assumptions, or a conceptual framework), while the adequacy of R to the reality being modeled cannot be established within R and needs an external validation R’, for which the story is repeated again. Such properties as safety are inherently probabilistic: only if a sufficiently big number of instances of a concrete product X is exploited for a sufficiently long period of time in sufficiently diverse environmental situations, can we then conclude that X is acceptably safe with enough confidence. Unfortunately, when a new system goes into production, neither of the above “if”s is available so that risk and uncertainty are inherited qualities of a new system. To manage them, classical engineering has always exercised care, incremental innovation and a reasonable conservatism (cf. [11]). However, the latter seems to be a difficult option at this time for autonomous driving, connected cars and similar initiatives, so we need a new approach somehow reconciling conservatism and novelty.

B. MAIN ASSUMPTIONS

Our approach to assurance can be seen as an integration of process- and product-oriented approaches. To make it doable and conservative, we rely on process-oriented foundations, but our notion of process builds aspects of product-assurance into the processes, by including relevant checks on product items using predefined acceptance criteria [13].

**Definition 1 (Main):** System X is considered to be acceptably safe (for a given application A in a given environment E) if the manufacturer’s safety engineering process, SEP, satisfies the following two conditions:

a) SEP is acceptably-well defined for a class of systems similar to X

b) SEP’s definition is acceptably-well executed for X, so that according to empirical evidence and expert opinion, if condition b) is satisfied for a system X, then X is safe for application A in environment E.

To make these general conditions technical and verifiable, we build three WF+-models: \( Wf_{SEP} \) that models the SEP, \( Wf_{\text{SEP}}(X) \) that specifies SEP’s execution for system X,

---

1 For example, the celebrated Gödel’s incompleteness theorem shows that even such a poor and seemingly formalizable universe as arithmetic cannot be formalized in a complete way.
and $Wf_{\text{Norm}}$ that models a body of normative documents (standards, regulations, on-site manuals, expert opinion and all relevant domain knowledge), which regulate and prescribe how SEP is to be defined but, simultaneously, give the manufacturer enough freedom to adapt to the norm. We specify these models as specially profiled class diagrams so that assurance amounts to checking if i) an object model (instance) conforms to the metamodel and ii) the metamodel is a correct refinement of another (normative) metamodel as schematically shown in the bottom block in Fig. 2. In the block arrows conf-i and conf-ii, black sub-arrows denote mappings between the underlying graphs while red arrows denote constraint satisfaction in conf-i and constraint weakening in conf-ii.

**Definition 2 (Main, more formal):** System $X$ is considered to be acceptably safe if the following two conformance conditions are satisfied:

$$Wf_{\text{SEP}}^\text{exe}(X) \triangleright^\text{inst} Wf_{\text{SEP}} \triangleright^\text{ref} Wf_{\text{Norm}}$$

where $\triangleright^\text{inst}$ denotes instanceOf-relation and $\triangleright^\text{ref}$ is a suitably defined refinement relation between workflows modelled as class diagrams. The index $\triangleright^V$ refers to the necessity to check both Validity of the data and Verification of the formal relation $\triangleright$; later we will explain how validity checks can be reduced to another layer of syntactic verification.

**Remark 1** For making the definition more precise and even formal, finding an appropriate notion of workflow refinement and its formalization are crucial; we leave this for future work. As a possible starting point, we are investigating the notion of model transformation network refinement proposed in [14].

**Remark 2** Workflow conformance includes not only control flow conformance (the primary approach in standard process-based approaches to certification) but also data conformance, which makes it product-based as well.

**Remark 3** In principle, the refinement chain can be extended with a deeper meta-norm $Wf_{\text{Metanorm}}$, regulating the definition of normative documents.

Thus, we reformulate assurance as a combination of (semi-)formalizable tasks, which are non-trivial, but known, partially well-studied and amenable to (semi-)automation. Using decomposition techniques described in [14], the conformance checks can be done in a hierarchical way, which would make them manageable even for very complex workflows.

### III. FROM GSN TO WF+ , 1: PROCESS & DATA FLOW

Figure 1 shows a GSN goal structure taken from the GSN Community Standard [6]; we will refer to it as the (GSN) Example. In this section, we will interpret the Example as a sketch of a simple safety case, extract its semantic meaning and present it in WF+ terms, and then compare the two cases, as schematically shown in Fig. 2. In more detail, we will extract from the data in the Example a SEP definition workflow $Wf_{\text{SEP}}$ and show that the argument flow of the Example can be interpreted as checking conformance $Wf_{\text{SEP}}^\text{exe}(X) \triangleright^\text{inst} Wf_{\text{SEP}}$, where $X$ denotes the Control System considered in the Example. We then somewhat artificially create a normative workflow to simulate the second conformance check. Thus, in this section all workflows occurring in formula $|$ are extracted from the data provided by the Example.

#### A. Recover Semantics: GSN through workflow glasses

Even a superficial inspection of the Example’s elements shows that they (directly or indirectly) refer to entities of several basic types. There are processes, e.g., context C4 mentions FHA (Functional Hazard Analysis), justification J1 mentions SIL Apportionment (SA), solution node Sn2 mentions FTA (Fault Tree Analysis), and goals G7, G8 talk about Protection Systems Development (PSD). There are also data, e.g., context C4 says that FHA produces a set of hazards, and strategy S1 together with goals G4-6 say that this set consists of three hazards, $H=\{H_1, H_2, H_3\}$, while goals G5-6 refer to probabilities of hazards $H_2$,$H_3$ occurrences. The results of such an inspection are shown in Fig. 4, where references to processes and data are labelled by green and orange block resp., and special data residing in normative documents, Tolerability Targets in C3 and SIL Guidelines in C5, are shown in grey (ignore blocks of other colours for a while). Data names are not specified to keep data blocks compact.

Colons in block labels show that we talk about instances of processes and data, i.e., individual executions of processes and individual data objects consumed and produced by processes. For example, we have an execution of some formal verification procedure FV, which takes as its input data about the Primary protection system (note the data block below and to the left of G7) and produces a boolean value T; the latter resides within node G4 and denotes the statement Elim($H_1$)=T, where Elim refers to the proposition specified in G4.

A more interesting dataflow is hidden behind goals G5-G6. Each of them assumes a separate FTA execution for the corresponding hazard (note two green FTA blocks), which inputs data about the corresponding protection system (two orange arrows going left-down) and outputs the probability of the hazard occurrence (two green arrows resulting in p2:D and p3:D). These probabilities are compared with the corresponding normative values, $10^{-6}$ for hazards $H_2$ having SIL=3 (see goal G7) and $10^{-5}$ for $H_3$ with SIL=2 (see goal G8); these normative values are taken from the Tolerability document. If the comparison returns True, we conclude that the corresponding hazard has been sufficiently mitigated (see goal G2 and strategy S1) and set Mit($H_i$)=T for i=2,3, where Mit denotes the italicized proposition above. We coloured the Mit-blocks differently from blocks T:B as the latter propositional statements are directly stated by the goal owning a red block while statements Mit($H_i$)=T required, besides reading goals G5-6, also i) reading elements G2, S1, C3, and ii) having some domain knowledge merging all information involved into a reasonable workflow. Domain knowledge is also needed to recover the dataflow between blocks FHA, SA, PSD and FV, FTA (we consider this in more detail in Sect. III-B and Fig. 4).

An important type of process is reviewing, shown in Fig. 5 by purple round blocks labelled R. For example, to assure the
Figure 1. A safety case example from the GSN Community Standard [9, p.14]

Figure 2. From GSN to WF+
statement required by justification J1, the process SA is to be reviewed, and similarly achieving goals G7 and G8 needs reviewing of the corresponding executions of PSD. A review R of process P (we omit colons) takes P’s execution data as its input and outputs a Boolean value for the corresponding proposition about P (typically, correctness and/or completeness phrased in context dependent terms, e.g., completeness of FHA as phrased in assumption A1). Note assumption A1 is not needed for strategy S1, but is necessary for deriving G1 from G2 and G3 (S0 is missing and was added by us).

The last ingredient of the story is logical inference: note light-blue blocks I residing in strategy nodes and light-blue arrows connecting it to propositions. The inference in strategy S0 can be specified by logical formula $A1 \Rightarrow (G2 \& G3 \Rightarrow G1)$, and similarly for the two other strategy nodes (S1 is unconditional while S2 requires a side condition J1). Although inference goes bottom-up, we show it with top-bottom arrows following the GSN notation to reduce clutter.

In the next subsection, we will convert our process-data annotations into an accurate workflow model, and then in Sect. IV will discuss the logical flow over the dataflow.

To simplify notation, we omit the corresponding data block and connect P and R immediately; an accurate workflow model will be considered later in Sect. III-D.

---

Figure 3. GSN Example annotated with data (orange & blocks), process executions (green & purple), logical propositions(red) and inference (pale-blue).

**B. Instance Conformance: Syntax**

Process and data blocks in Fig. 3 sketch some workflow, which in this section we treat as an instance of some workflow definition/metamodel ($WF_{SEP}$ in term of Sect. II). Making the latter syntactically accurate and semantically reasonable needs some domain knowledge beyond the workflow sketch in Fig. 3, but we tried to minimize the extension beyond the Example.

We build the metamodel as a class diagram with a special type of class stereotyped Process (green in Fig. 4), whose instances are process executions. Dataflow to and from processes is shown with directed associations, while ordinary (static) associations between data classes (yellow and orange) are all undirected and oriented vertically. Orange data are input to the entire SEP process, consisting of three components, HARA, Design, Verification (note three green rectangles in Fig. 4), which are further decomposed as shown. The order of processes is shown with numbers prefixing process names (and can be inferred from the dataflow). We use rectangle frames to show decomposition, e.g., process HARA consists of three subprocesses. Boundary objects (circle with a vertical line attached) are used as aliases for data to reduce the number of crossing lines in the model, and purple elements will be discussed later (ignore them for a while). Red notes are used to denote constraints on data. For example, the attribute valid? of SIL must be True. SysML-like ports are used on the sides
Figure 4. Metamodel further referred to as $W_{SEP}$.
of classes to indicate that an attribute is produced by a process. Attribute indentation denotes nesting (attributes of attributes).

The diagram is mainly a self-explainable completion of the annotated Example (Fig. 3). Black-box labels loosely show the backward mapping to the GSN diagram to illustrate how our metamodel was derived from it. We only mention several points. Note attribute mitORElim of class Hazard and its multiplicity [1], which says that each hazard is to be eliminated or mitigated but not both. If a hazard is to be eliminated, then by composing associations occursIn and protectedBy, we obtain a derived association Elimination of multiplicity 0..1 as not each hazard is to be eliminated; similarly for hazards to be mitigated we obtain a derived association Mitigation. Now multiplicity [1] implied multiplicity [1] spanning the two association ends (shown blue as it is derived) says that each hazard is to be eliminated or mitigated by exactly one PS. The inverse multiplicity [1..*] is derived from multiplicities of protecs and consistsOF and demands a protection system to protect against at least one hazard (this kind of association is not essential for safety, but is useful for ensuring there are no idle systems not protecting hazards, for example, after a design iteration). Also, process FV in Fig. 3 is specialized as a state space exploration, which assures hazard elimination if the hazardous state is not reachable. The comparisons residing in goals G5, G6 are specified as executions of the query specified in box 3b.2, which is defined to assure hazard mitigation.

Note a special feature of the workflow: data produced by a process are connected to its input data, e.g., Process 3b.1 takes a pair (H,PS) consisting of a hazard H and a PS intended to mitigate H, uses this data to conduct a fault tree analysis, and outputs the Real Probability of H occurring with PS in place as an attribute of H.

The metamodel in Fig. 4 defines a workflow, which can be executed for any data correctly instantiating the orange part of the diagram, and thus instantiate the yellow part of the class diagram (commonly referred to as WorkProducts). The GSN Example actually tells us a story about such an instance \( Wf^{sep}_{\text{recon}}(X) \) as shown in Fig. 3 where the dataflow from (green) processes to (yellow) are omitted to avoid clutter (ignore purple underlined circles with arrows for a while). It is easy to see that the instance workflow \( Wf^{recon}_{\text{SEP}}(X) \) satisfies all constraints declared in the metamodel workflow \( Wf_{\text{SEP}} \).

C. Instance conformance: Semantic validity

In assurance, there are two main questions about instances. The first is its syntactic correctness: whether all constraints declared in the metamodel such as \( Wf_{\text{SEP}} \) in Fig. 4 are satisfied. This task can be done automatically with modern MDE tools. The second question is more challenging: syntactically correct data can be semantically invalid e.g., for instance \( Wf^{recon}_{\text{SEP}}(X) \), a hazard can be missing from set Hazard.allInstances=\{H1, H2, H3\}, or the SIL of hazard H3 is to be 4 rather than 2 so that hazard grouping must be changed, or if even all SILs are valid, grouping was done incorrectly and hazard H2 is to be grouped with H3 rather than H1, etc. In fact, a majority of the instance elements are to be checked for semantic validity (we will say validated), including the results of verification. Validation of an element \( x \) in instance \( Wf^{sep}_{\text{SEP}}(X) \) is done by reviewing the process that produced \( x \), e.g., FHA for the hazard set completeness, SIL Determination for SILs validity etc.. Review processes (we say reviews) are denoted by purple underlined circles in the metamodel Fig. 4; we only show three of the total six (not seven as Query is computed automatically and does not need reviewing). Review executions in Fig. 5 are also purple. To simplify notation, we show the dataflow from a process to its review by an arrow and omit the intermediate data: this is just a syntactic sugar for a detailed specification considered in the next subsection.

D. Reviewing as an aspect

The pattern in Fig. 5 of a process producing data followed by a review checking the validity of that data is shown only once but in reality is repeated in many different places in a typical SC. Thus, reviewing appears as a typical crosscutting concern and should be appropriately managed along the lines developed by the aspect-oriented programming community (AOSD), [15]. In our context, we need a mechanism of enriching a core metamodel with an aspect metamodel (advice, in AOSD terms) developed separately and weaved into the core metamodel at specified points (pointcuts). We are not aware of any AO-frameworks for metamodeling presented in the literature, and leave development of such for future work. Below we will use the main idea of aspect weaving [16] for metamodel weaving in the following sense.

A crosscutting concern is specified by an advice metamodel \( M_{\text{adv}} \), whose part \( M_{\text{entry}} \subseteq M_{\text{adv}} \) specifies an entry, to which the advice is to be applied. Formally, we model this as an embedding mapping \( e : M_{\text{entry}} \rightarrow M_{\text{adv}} \). Now, having a main metamodel \( M_{\text{main}} \) with a set of entry points in it, for each such point \( e \) we specify a mapping \( w_{\text{ep}} : M_{\text{main}} \rightarrow M_{\text{entry}} \) and apply the pushout operation (PO) to the pair \((e, \! w_{\text{ep}})\), which will result in the merge \( M_{\text{main}} + M_{\text{adv}} \) modulo the common (shared) part \( M_{\text{entry}} \); thus, we have weaved \( M_{\text{adv}} \) into \( M_{\text{main}} \).

In Fig. 4 three (from total six) entry points are shown with underlined purple circles, and the advice metamodel to be weaved with them is described in Fig. 4. It shows a very simple example of an actual review process around the key idea: the review observes performance data of a process, say P, including when and how it was executed and who the executor(s) were, alongside with the inputs and outputs of P. The review then outputs a Boolean value that (dis)approves the idea: the review observes performance data of a process, say P, including when and how it was executed and who the executor(s) were, alongside with the inputs and outputs of P. What data are to be validated, and how reviewing is to be organized, is to be determined in the normative documents and be accordingly implemented in SEP.
Figure 5. An WF$^+$-instance further referred as $Wf_{\text{SEP}}^+$

IV. FROM GSN TO WORKFLOW+, 2: ARGUMENT FLOW

In Sect. IV-A, we show how the data-driven inference mechanism of WF$. We will try to follow the argument flow of the Example as much as possible, but be forced to deviate from it: the argument flow in the Example does not match its data flow, whereas our formal data-driven inference procedure forces the former flow to follow the latter. We discuss the discrepancy in Sect. IV-C after we show how inference on the level of metamodel interacts with their instances in Sect. IV-B.

A. Data-driven logical inference

The WF$^+$-based inference is described by Fig. 7. It shows the metamodel $Wf_{\text{SEP}}$, in which green process blocks are omitted to save space (but implicitly are there) but aliases (in fact, entry points) of all six review processes are shown (in purple). There are several changes in the representation of the data part. We introduce an abstract reified association $hzProtection$ (generalizing associations Elimination and Mitigation (whose lines are omitted to avoid clutter), and derive from multiplicities for attribute mitORelim and associations Containment (Cnt) and grProtection (see Fig. 4) that $hzPr$ is composed

$$hzProtection = \text{Containment} \gg \text{grProtection}$$

with multiplicities as shown (circled blue as they are derived). To shorten formulas, we will abbreviate the names above by $hPr$, Cnt, and gPr resp. The fact that association $hPr$ is composed from Cnt and gPr, is shown in the diagram by two orange arrows from the component associations to the composed one——this is a simple case of data derivation (or querying). Important new elements in Fig. 7 are claims, i.e., natural-language formulations (in the assurance parlance) of data constraints, which are shown as notes attached to constraints. E.g., claims A11, A21 (pink notes on the left) are attached to encircled data constraints as shown, and claims A12, A22 (bold red and white font) are attached to validity constraints. Note an important difference between pink and red claims. Claim A11 is a simple syntactic constraints violated if a hazard $H$ does not have a SIL (all attributes have multiplicity 1 by default) or its SIL is an integer greater than 4. Claim A12 is a semantic requirement of the validity of SIL assignment,
e.g., if a hazard H is assigned SIL=2 when it is so dangerous that its SIL should be 4, then constraint A11 holds while claim A12 is violated. To formalize the latter as a constraint, we introduce SIL’s attribute valid of type Boolean, produced by the corresponding review process, and require the value of this attribute to be T(ue). In GSN, conjunction of (syntactic) A11 and (semantic) A12 would typically be formulated in semantic terms as phrased in the claim note A1. In a similar way, we can treat claim A2 (below A1) as conjunction of a syntactic part A21 (the pink note) phrasing multiplicity 1 (note the dashed line), and a semantic part A22 validating whether the hazard grouping is done right w.r.t. safety engineering guidelines. Conjunction of A1 and A2 can be formulated as in the claim note A so that we have the inference tree from A11, A12, A21 to A as shown in the diagram. We can identify claim A with context J1 of the Example, more accurately, constraint A can be seen as a formal model of (somewhat fuzzy) claim J1 (note the black box labelled J1 attached to note A). Based on this assumption, we somewhat arbitrarily distribute the meaning of J1 between its subclaims (note labels 1/4 of J1).

Our seemingly pure logical work above was actually guided by data and dataflow in the metamodel. Two orange (the data colour) dashed lines from note A to class SIL (data element D1(A)) and association Containment (element D2(A)) which point to data defining the validity of claim A. The meaning of A then guides us in finding the relevant data constraints A11 and A21, while processes that produce data D1(A), D2(A) point to reviews needed for validation—the decomposition of A into conjunction $\bigwedge_{1 \leq i \leq 2} A_{ij}$ is data driven!

We can proceed in exactly the same way with decomposition of claim GP (below claim A that refers to the Apportionment part of the story while GP stands for the GroupProtection part). We will mention briefly several points. The semantic claim GP2 is formulated in parallel to goal G2 in the example, and the name of the semantic validity attribute is changed from the common valid to a special name following the Example’s context C5. Then we infer

$$GP \Leftarrow GP1 \land GP2, \text{HC} \Leftarrow A \land GP,$$

where HC (HazardCoverage) is a claim about the composed association hPr (note the orange link). We also know that the SEP requires to verify all hazard coverage links (which instantiate association hPr). Verification processes evaluate attributes ver? (whose Truth is required by corresponding pink constraints), which are then validated by reviews evaluating attributes valid? (with red Truth constraints). We thus have an inference tree for claim CV as shown in the diagram. Next
we assume that having a coverage link H-PS between hazard H and system PS (an instance of hPr) verified and validated means that system PS does protect against H. We then derive claim HP. Finally, assuming hazard analysis completeness HI, we derive the top safety claim SS as shown.

B. Constraint derivation: From metamodels to instances

Constraints are elements of metamodels, and their derivation lives on the metamodel level while a typical GSN case builds an argument flow for a concrete system, i.e., an instance. The gap is easily bridged by the soundness of our derivation steps: we derive semantically rather than by syntactic rules. In more detail, suppose we have a metamodel $M = (G, C_M)$ with $G$ being its graph of classes and associations, and $C_M$ a set of constraints defined over $G$. An inference tree can be schematically presented by a sequence of inference steps

$$C_0^{\text{given}} \models C_{1}^{\text{der}} \models \ldots \models C_{n}^{\text{der}} \models C_{n+1}^{\text{der}} \models C^{\text{der}} \models C! \quad (4)$$

where $\models_i$ is an inference step extending a set of derived constraints $C_{i-1}^{\text{der}}$ with a new constraint $C_i$ so that $C_{i}^{\text{der}} = C_{i-1}^{\text{der}} \cup \{C_i\}$; the initial set consists of all constraints initially given in the metamodel; and $C^{\text{der}}$ is the final set containing the top claim $C!$. Soundness of a step $i$ means that for any instance $I$ properly typed over $G_M$, if $I \models C_{i-1}^{\text{der}}$ then $I \models C_i^{\text{der}}$. By transitivity, if $I \models C_0^{\text{given}}$ and all steps are sound, then $I \models C!$.

Soundness of some of our derivation steps is based on simple logic, e.g., the inference $CV \Leftarrow EV \& MV$ is sound because of the data condition $hzProtection = \text{Elimination} \cup \text{Mitigation}$, which is in turn implied by multiplicity 1 of Hazard’s attribute $\text{elimOrMit}$ (see Fig. [3]). Some inference steps can be seen as merely definitions, e.g., $A1 \Leftarrow A11 \& A12$. And some derivations
are semantically non-trivial, e.g., when we conclude that having a coverage link H-PS both verified and reviewed implies that PS really protects against H. A common approach to such derivations is to require strong validation criteria so that getting equality valid=≡T for an instance I is not easy but being achieved would indeed mean that the semantic condition in question is satisfied.

Thus, for a given system X, if $W_f^{\text{exe}}(X) \models C^{W_f^{\text{sep}}}$, then $W_f^{\text{sep}}(X) \models C^!$ so that the B part of an WF+-based safety case (see Def. 1) consists of two major subparts:

(B1) Constraint derivation over the SEP metamodel, $C^{W_f^{\text{sep}}}$ $\models C^!$, and

(B2) Checking instance conformance, $W_f^{\text{exe}}(X) \models C^{W_f^{\text{sep}}}$, for a given system X, whose most challenging part is checking T for validity constraints (shown in red in Fig. 7).

V. WF+ vs. GSN: A COMPARATIVE ANALYSIS

We will begin with a brief survey of how we processed the material described in the Example (further Ex.GSN) and come to its WF+-version (Ex.WF+) to set a stage for the discussion, then present the results of comparing Ex.GSN and Ex.WF+, and conclude with some general observations about GSN.

A. From Ex.GSN to Ex.WF+: Defining the DCC

We recognize that the main goal of the Ex.GSN diagram was to demonstrate the vocabulary of GSN elements and show how they can usefully be put together rather than write an SC. To do this, they chose to specify in GSN terms an oversimplified SC, which a fully justifiable and reasonable choice: GSN is widely used for building and documenting SCs in many domains. The Ex.GSN-authors addressed both facets: they describe the notation and skilfully managed to present an interesting and seemingly well arranged SC within a small diagram. The fact that Ex.GSN has resided in the Standard without changes for nine years now justifies this conclusion and motivates our idea to use Ex.GSN as a sample SC to show how WF+ works. For our comparative analysis, we need an accurate description of what our processing brought to Ex.GSN.

As Fig. 1 shows, Ex.GSN provides a listing of processes and data involved, and Fig. 4 shows that we did not introduce in Ex.WF+ new processes nor data classes (they all are labelled by the respective elements of Ex.GSN). However, classes (green and yellow) only give us a discrete view of the story without the dataflow between processes and data (green-yellow links), and associations between classes (yellow-yellow); below we refer to both types of links as dataflow. We have restored the missing structural links based on our understanding of the domain knowledge (DK) and thus came to metamodel $W_f^{\text{sep}}$ in Fig. 4. As this DK is sufficiently common, it seems reasonable to assume that the authors of Ex.GSN based their SC on a similar dataflow picture but kept it implicit. In addition, our WF+-glasses revealed two more actors on the stage: a detailed specification of constraints for the dataflow, and splitting the argument flow into two parts: (data-driven) constraint derivation over the metamodel, and checking the instance conformance $W_f^{\text{exe}}(X) \models_{\text{inst}} W_f^{\text{sep}}$ (while a typical GSN SC considers argument flow for a given system X without mentioning the two levels).

We will refer to the differences between the two frameworks, GSN and WF+, as to DCC (Dataflow, Constraints, Conformance). Again, it seems reasonable to assume that all three actors are implicitly (and perhaps inaccurately) preset in the Ex.GSN so that the only real difference between the two frameworks is that WF+ treats DCC as first-class citizens while GSN keeps them underground. We can schematically write something like an equation

$$WF^+ = GSN + \text{Explicit DCC.}$$

Our next goal is to compare the two cases, Ex.GSN and Ex.WF+, to see whether DCC-explication is practically important or is mainly a theoretical bonus.

B. Mapping between the two diagrams

We compare two argument flow diagrams: GSN-diagram in Fig. 4 and WF+-diagram in Fig. 4. In the latter, note black labels that map elements of WF+-diagram (we will say ‘ours’) to elements of Ex.GSN (theirs). Some elements are a perfect match, e.g., our claims A and HP are exactly their justification J1 and goal G2 resp.; the labels can be seen as a wf2gsn-relation between the two diagrams. With this discipline, some of our claims have two labels (fraction labels use different colours to ease visualization), which makes relation wf2gsn of one-to-many type. Finally, several elements in the lower part of our diagram are labelled X# to be read as “the type of X”. This is necessary when Ex.GSN refers to inference level data (e.g. Sn3#: a PS is developed according to the correct SIL guidelines) that must be modelled in the SEP (e.g. GP2: PS must be developed according to the correct SIL guidelines).

Now some of our diagram claims have several labels, e.g., GP2 has label 1/8ofG2 by its place in our inference tree, label G3 as it is almost exactly Ex.GSN’s G3, labels G7#,G8# because conformance of an instance $W_f^{\text{sep}}(X)$ to GP2 would correspond to Ex.GSN’s goals G7, G8 and solutions Sn3, S4 resp. Having placed all labels and hence defined the relation wf2gsn, we can begin our comparative analysis.

Mapping wf2gsn maps claims (nodes) in WF+-diagram to elements/nodes in GSN-diagram. For an ideal match between two diagrams, we would like this mapping to be compatible with the inference structure: if two our claims C1, C2 are mapped to GSN-elements E1, E2, and C1 is a subclaim of C2, then E1 is a subclaim of E2 if they are both claims (this is an ideal case), or at least E1 is a “subelement” of E2 in some suitable sense, e.g., an assumption attached to a strategy S below goal G can be seen as a subelement of G. If the subelement relation is preserved but its depth is not, e.g., C1 is an immediate subgoal of C2 while E1 is a grandchild subgoal of E2 (or the other way round), it would mean that one of the diagrams have a more detailed inference path than the other, which is still a good match between inference systems.

Surprisingly, our relation wf2gsn is not compatible with the inference structure and actually significantly distorts it.
The most visible, and dramatic, distortion is the clear parallel structure of GSN-diagram: below goal G1, the flow splits into two parallel branches below G2 and below G3, while in the WF+-diagram-image of G3 is claim GP2 and thus the G3-branch is embedded into the G2-branch. Another surprising distortion is placement of assumption A1 as a subelement of G2 while actually it must be subelement of the top SS in WF+-diagram or G1 in GSN-diagram (see Fig. 2 in which we introduced a strategy S0 to which A1 actually belongs); this latter placement of A1 is logically obvious. There are also smaller discrepancies such as having context C3 in the upper part of GSN-diagram, while its proper place is close to bottom as shown by WF+-diagram. Another major distortion issue is non-distinguishing of the instance-metamodel layers in GSN. E.g., in GSN-diagram, having a common Sn2 for two goals G5, G6 is correct on the level of types (see WF+-diagram) but incorrect on the level of instances: it may happen that FTA was executed correctly for hazard H2 and incorrectly for H3, necessitating different reviews and thus two executions of Sn2 as shown in our instance diagram Fig. 5.

C. Discussion

1) WF+ in the GSN view: The GSN literature identifies the two basic ingredients of safety assurance: deduction (of derived claims from the elementary claims until the top claim is reached) and evidence (that supports the elementary claims) [7]. On the conceptual level, we can identify the task (B1) (at the end of Sect.IV) with GSN’s deduction, and task (B2) with the evidence support: indeed, checking an instance conformance for a typical WfSEP basically amounts to checking non-empty instantiation of the type graph GWFSEP that satisfies multiplicity constraints and checking T for validity attributes, which is close to providing evidence in GSN.

2) GSN in the WF+ view:

a) The GSN approach to structuring seems ad hoc. In WF+, the inference tree is built from claims (constraints) while GSN’s inference involves several types of elements (note that claims in Fig. 4 are labelled by G, A, J). Analysis of labelling in Fig. 4 shows a close relationship between data and context, but other types also appear there.

b) While in WF+, GBOO data drive the inference and thus facilitate SC building, GSN leaves the SC builder on their own with the problems of proof and evidence design, which can lead to an ad hoc inference structure as we have seen above.

c) GSN flattens the two-level structure of checking the SEP execution: constraint derivation (B1) and instance conformance (B2). Flattening often hampers comprehensibility and creates errors (even our simple comparisons revealed issues), which can dangerously accumulate in large industrial safety cases.

d) The absence of formal semantics and precise usage guidelines for GSN elements contributes to all issues above.

e) Perhaps the most serious issue is the absence of explicit traceability mechanisms (cf. discussion in Sect. VII).

VI. WORK IN PROGRESS AND FUTURE WORK

A. Conformance to Normative Doc

1) The Normative Document: A difficult but essential question remains: how do we know the SEP is acceptably well defined (cf. Def. 1h) and will result in a safe system if properly executed? Paper [11] outlines how this issue is addressed in the classical engineering domains. Similar to our requirement R6, one of then main points is that sufficiently prescriptive guidelines based on established foundations (we say ‘normative documents’) for the devlopment of safety-critical systems must be provided to ensure their safety.

In WF+, we propose to model the integrity of normative documents (domain-specific standards, internal company procedures, legal requirements) as a metamodel WfNorm to guide the development of a WfSEP and, moreover, to provide a definitive second layer of assurance by checking conformance WfSEP ⊩ref WfNorm (item a) in Def. [6] A manufacturer would develop their WfSEP by refining a very abstract WfNorm to the point that it can be executed. Through conformance to WfNorm, a manufacturer can be sure that they satisfy all prescriptive guidelines and thus gain confidence in the safety of their system (as long as WfNorm is well defined).

Conformance in this sense has two components. a) Syntactic conformance is shown by demonstrating that a SEP metamodel is indeed a refinement of the normative metamodel. b) Semantic conformance, which is up to reviewers - they must use their knowledge to determine if the SEP satisfies the intent of the normative metamodel. The first component of conformance is directly formalizable by a suitable notion of refinement, but the second is more difficult and is currently a work in progress. Fig. 8 serves as an example to illustrate this concept (is far too simple to provide any meaningful guidance to manufacturers implementing it or reviewers checking an implementation). Traditionally, standards are written in natural language, contain many cross referenced sections, subsections, tables, etc., and can require a huge amount of time and effort to understand fully. Even once someone has spent enough time with a standard to understand it fully, their interpretation may differ from someone else with the same experience due to the ambiguities of natural language and disjoint representation of information.

From a MDE perspective, replacing the traditional approach with a model-based WF+ could provide the following benefits: a) it avoids the ambiguity of natural language b) it improves understandability by providing explicit process flow, dataflow, and constraints c) it presents information more intuitively by utilizing block diagram notations.

2) Multiple Normative Documents: Manufacturers normally must conform to many domain-specific standards, internal company procedures, legal requirements, and other normative documents. These standards are often interrelated, overlap and can even be contradictory. The WF+-approach models this as a single virtual workflow defined as an imaginary merge of the components, in which discrepancies are reconciled. Implementation of such a merge requires an accurate specification of all complex relationships between normative documents,
3) Reasoning and Justification: As explained in Sect. IV, establishing the soundness of an argument may be difficult. While an instance must only demonstrate conformance to its SEP, and an SEP must only demonstrate conformance to its normative document, a normative document must also be justified somehow. This justification can be interpreted as one final conformance mapping to a system of informal elements based on experience, empirical evidence, best practices, and patterns.

B. Something about a metamodel for WF+.

WF+ is a well-defined formalism, but we did not include its metamodel into the paper due to space limits. It also has an intuitive semantics: a WF+-process P can be seen as a model transformation (MT) either N ports, each with a metamodel assigned. Given N instances over those metamodels, P transforms them into an instance over some output-port metamodel. Importantly, input and output metamodel may overlap (actually this is a typical case for SEPs we are dealing with), which adds + to the name WF+. An accurate, and manageable, formalization of WF+-semantics needs category theory and is a work in progress.

VII. APPLICABILITY AND POSSIBLE IMPACT

We have analyzed seven major issues with SCs identified in a survey on SC use in practice [12]. These issues correspond well with our own experience of safety cases in industry and overlap with our requirements described on p.1. We provide an italicized summary of issues identified in the survey and a reference to our similar requirements, followed by the applicability of WF+ in corroborating and/or addressing the issue, followed by GSN’s very limited ability to manage or, worse, its ability to create the issue.

1) Scalability: Navigation and comprehension of the SC is difficult. Current tools either fail to deliver features to handle scalability, or are domain-specific and not generalizable. (Plus R4, R5)

WF+: a) All traceability is explicit and extends into the development process, the product artifacts and even the environment, thereby improving detailed understandability, b) The framework is model-based and allows for compositionality and hence decomposition and modularization and so, in spite of the complexity of the system and its trace-links, there are built in constructs for managing this.

GSN: SCs are difficult to comprehend due to the amount of implicit information and their ad hoc structure that is evident even in simple SCs (see Fig. 3); for large industrial SCs, these deficiencies can accumulate enormously and really prevent understanding.

2) Managing Change: There are no effective mechanisms to manage changes in SCs. Lack of integration of processes and inadequate tool support are contributing factors. (R1, R2, R5)

WF+: The SEP and SC meta-models are integrated. We thus have complete traceability between model elements, explicit dataflow and explicit process flow, all of which are crucial for
impact analysis and change management. A strict separation of instance and meta-levels of modelling provides a convenient reusability mechanism.

**GSN:** Traceability is almost entirely omitted, making change management a very difficult task. The GSN SC is a completely separate entity from the SEP and other product artifacts.

3) **Requiring special skills to create:** Graphic notations of SCs are typically intuitive to understand, but creating a well-structured, convincing SC requires special skills and experience. The privacy of information in SCs prevents the sharing of knowledge on effective patterns and strategies. (R3: a unified approach reduces the need for special skills.)

**WF+:** a) Workflows are a natural way for engineers to understand their work. As a result, WF+ provides an intuitive and easily understandable framework for building SCs. b) A strict separation of instance and meta- (and metameta-) levels of modelling allows experts to create templates and share effective patterns and strategies (with a reduced fear of revealing sensitive information about products).

**GSN:** For engineers, it is difficult to convert their intuitive understanding of the story of a system’s safety into the one-dimensional structure of GSN-style safety cases.

4) **Complexity of the system:** Identifying safety concerns is extremely difficult in complex systems due to the difficulty of managing and identifying interactions between different subsystems. Multidisciplinary collaboration is also challenging. (R2, R4, R5)

Complexity. **WF+:** a) Traceability does help to identify interactions. b) Compositionality of WF+ allows for decomposition a classical engineering approach to manage complexity.

**GSN:** The absence of traceability links makes interaction analysis very difficult unless the SC assessor has been deeply involved in the creation of this system. But then the assessment can suffer from confirmation bias.

Collaboration. **WF+:** An intuitive common language allows experts to share different perspectives and facilitate discussions

**GSN:** Ad-hoc structure and implicit information leaves room for ambiguity, potentially impeding communication between teams.

5) **Uncertainty, trust, confidence:** Issues of confidence and trust are seen as intangible. This leaves room for doubt about the efficacy of SCs and makes it difficult to establish trust. (R3, R4, R6)

**WF+** and trust are notoriously difficult to quantify in the context of assurance, and have long been a subject of debate [13]. **WF+:** a) Conformance of the SEP-definition to normative documents makes a “half” of the framework and thus directly addresses the issues above. b) Creating data-driven argument flow in WF+ is difficult, but if managed, it provides a high level of confidence as it is explicit and formalized.

**GSN:** The ability of manufacturers to generate trust in their safety cases is dependent upon their ability to express what has been done to reviewers. GSN-style SCs provide little support for this as they leave massive arrays of data, process and dataflow details implicit.

6) **Too flexible:** Flexibility can be dangerous because manufacturers can overlook aspects of safety. Conflicts of interest and confirmation bias contribute to this danger. This is closely related to establishing trust and confidence. (R6)

Independent assessment has been shown to be effective in reducing confirmation bias [19]. **WF+** provides a well-defined structure for SCs. A well-defined structure for SCs and the ability to offer templates to guide their development, allowing assessors to gain expertise in evaluating SCs.

**GSN:** The ad-hoc structure of GSN-style safety cases makes it difficult to develop expertise in analyzing these safety cases, making it far too likely that holes in arguments will go unnoticed.

7) **Incomplete information:** Survey: Limited integration of SC management into the software development process causes challenges in collecting sufficient and accurate information for safety arguments. Incomplete traceability, insufficiencies in testing and flawed safety requirements are all issues that arise as a result of this. (R3, R4, R5)

**WF+** processes and complete traceability between data are incorporated. This enables integration of the Design and SEP and SC development, allowing for systematic change impact analysis and, with proper tooling, automated updating of SCs in response to changes in data and gathering of data, ensuring information is properly managed.

**GSN:** a) GSN-style safety cases argue over instance-level data. The notion of processes flow and dataflow are omitted almost entirely. b) Implicit traceability leaves room for data to be left behind or forgotten.

VIII. RELATED WORK

A series of seminal papers [20–24] introduced the ideas of metamodeling and model management into assurance, but their main focus was on data conformance rather than dataflow. Dataflow was, indeed the focus in [13] but they do not consider the argument flow and focus on process decomposition rather than on SCs. Analysis of GSN-based SCs has an extensive literature, but it is mainly focused on the logic rather than the dataflow. A survey and many references can be found in Rushby [17]. Several ways of formalizing SCs were suggested by Rushby in [23], but his focus is more on the argument structure. Tooling for GSN-based SCs is discussed in [13].

**WF+** has its origin in block diagrams (as a networking mechanism) and model transformations as processes over GBOO data, but differs from both of them as illustrated by the inset schema. Model refinement [24] and process refinement [15] have extensive literature, but we have not found yet the notion most suitable for our needs to model conformance to the normative documents.

IX. CONCLUSION

We have presented **WF+** as a modelling framework which meets the core needs of creating, documenting and managing
SCs. Precise dataflow modelling, the benefits of the trace-ability inherent in UML, the intuitive understandability of workflows, and a well-founded approach to developing assurance assets, results in an approach that integrates processes, product and SC. This signifies a crucial distinction from GSN-based assurance cases. In our view, keeping the enormous dataflow part of assuring complex systems implicit, as well as separating the SC and assurance and development processes is as dangerous as neglecting the underwater part of an iceberg. Hence our significant departure from GSN-like SCs. We also showed that we can generate a view that resembles a GSN-like SC, and in this way the reader can appreciate what is missing in GSN that is inherent in WF$^+$. 
REFERENCES


