Combining Software Evidence – Arguments and Assurance

Rob Weaver Georgios Despotou Tim Kelly John McDermid

Department of Computer Science
University of York, UK
{rob.weaver, george, tim.kelly, john.mcdermid}@cs.york.ac.uk

ABSTRACT

Argumentation is an approach which can be used for describing how evidence satisfies requirements and objectives. A structured argumentation notation allows developers to determine the need for individual items of evidence and allows reviewers to determine whether a complete set of evidence satisfies the requirements. This paper introduces an established argumentation notation from the safety critical domain, as well as new research into dependability arguments and assurance of arguments. These techniques and concepts have been applied to the development and certification of safety critical software and it is believed they are both applicable to and beneficial for the wider field of evidence-based software engineering.

General Terms

Measurement, Documentation, Design, Verification.

Keywords

Argumentation, Safety, Requirements, Evidence.

1. INTRODUCTION

Argumentation is commonly used during safety critical systems development to demonstrate that system safety requirements are met. These arguments must be supported by evidence, and thus industry best practice follows an evidence-based approach to safety critical systems development. The argument creation process provides a technique for identifying what evidence needs to be produced about a system or process. The Goal Structuring Notation is a tested approach for constructing safety arguments. This paper begins with an introduction to the Goal Structuring Notation, used in Safety Critical industries as part of the development process for both software and hardware. It then discusses how the notation could be applied to evidence-based software engineering. Finally, current research into dependability arguments and assurance of arguments is presented, which has an impact on the use of GSN for evidence-based software engineering.

REBSE'05, May 17, 2005, St. Louis, Missouri, USA.

2. SAFETY CASES

Many modern safety critical standards (e.g. the UK MoD Defence Standard 00-56 [1]) advocate an evidence-based approach towards the development and certification of safety-critical software. The current trend in safety critical software development shows a move away from prescriptive process-based standards towards the use of a safety case with evidence specifically tailored to the system and software developed.

The purpose of a safety case can be defined in the following terms: A safety case should communicate a clear, comprehensive and defensible argument that a system is acceptably safe to operate in a particular context. The concept of the 'safety case' has already been adopted across many safety critical industries (including defence, aerospace, nuclear and railways). A safety case consists of three principal elements: Requirements, Argument and Evidence. The relationship between these three elements is depicted in Figure 1.

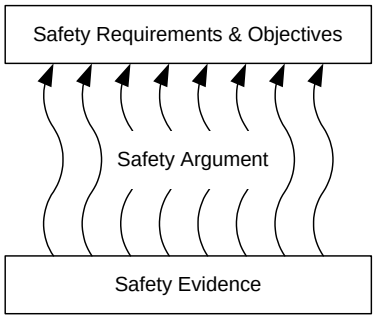

Figure 1 – The role of argumentation

The safety argument is that which communicates the relationship between the evidence and objectives. Both argument and evidence are crucial elements of the safety case that must go hand-in-hand. Argument without supporting evidence is unfounded, and therefore unconvincing. Evidence without argument is unexplained – it can be unclear that (or how) safety objectives have been satisfied.

3. THE GOAL STRUCTURING NOTATION

The Goal Structuring Notation (GSN) [2] – a graphical argumentation notation – explicitly represents the individual elements of any safety argument (requirements, claims, evidence and context) and (perhaps more significantly) the relationships that exist between these elements (i.e. how individual requirements are supported by specific claims, how claims are supported by evidence and the assumed context that is defined for

the argument). The principal symbols of the notation are shown in Figure 2 (with example instances of each concept).

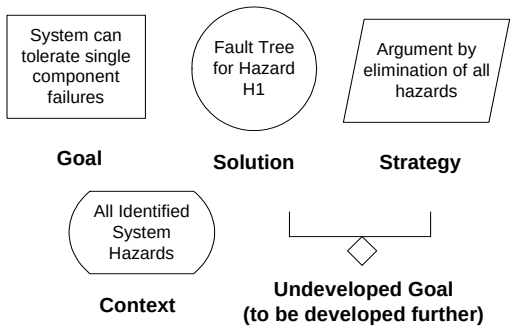

Figure 2 – Principal elements of the Goal Structuring Notation

Within Europe, GSN has been adopted by a growing number of companies within safety-critical industries (such as aerospace, railways and defence) for the presentation of safety arguments within safety cases.

When the elements of the GSN are linked together in a network they are described as a '*goal structure*'. The principal purpose of any goal structure is to show how goals (claims about the system) are successively broken down into sub-goals until a point is reached where claims can be supported by direct reference to available evidence (solutions). As part of this decomposition, using the GSN it is also possible to make clear the argument strategies adopted (e.g. adopting a quantitative or qualitative approach), the rationale for the approach and, importantly, the context in which goals are stated (e.g. the system scope or the assumed operational role). This allows the documentation of how evidence is combined and the context in which the evidence is applicable. It also allows the combination of a variety of different types of evidence.

Figure 3 shows an example goal structure. In this structure, as in most, there exist 'top level' goals – statements that the goal structure is designed to support. In this case, "*C/S (Control System) Logic is fault free*", is the (singular) top level goal. Beneath the top level goal or goals, the structure is broken down into sub-goals, either directly or, as in this case, indirectly through a strategy. The two argument strategies put forward as a means of addressing the top level goal in Figure 3 *are* "*Argument by satisfaction of all C/S (Control System) safety requirements*", and, "*Argument by omission of all identified software hazards*". These strategies are then substantiated by five sub-goals. At some stage in a goal structure, a goal statement is put forward that need not be broken down and can be clearly supported by reference to some evidence. In this case, the goal "*Unintended Closing of press after PoNR (Point of No Return) can only occur as a result of component failure*", is supported by direct reference to the solutions (items of evidence), "*Fault tree cutsets …*" and "*Hazard Directed Testing Results*".

4. GSN FOR SOFTWARE ENGINEERING

Primarily the focus of GSN in the safety critical industry is to provide a structured argument as part of a safety case to allow the assessment of system safety. However, the notation is used throughout the system development lifecycle, rather than just as an activity at the end of development. Developing a goal structure from high level requirements during the system development lifecycle allows the low-level requirements for evidence to be identified. Thus, the need for testing, analysis and other evidence generation approaches can be determined during system design. The safety argument can be used as a spine for the

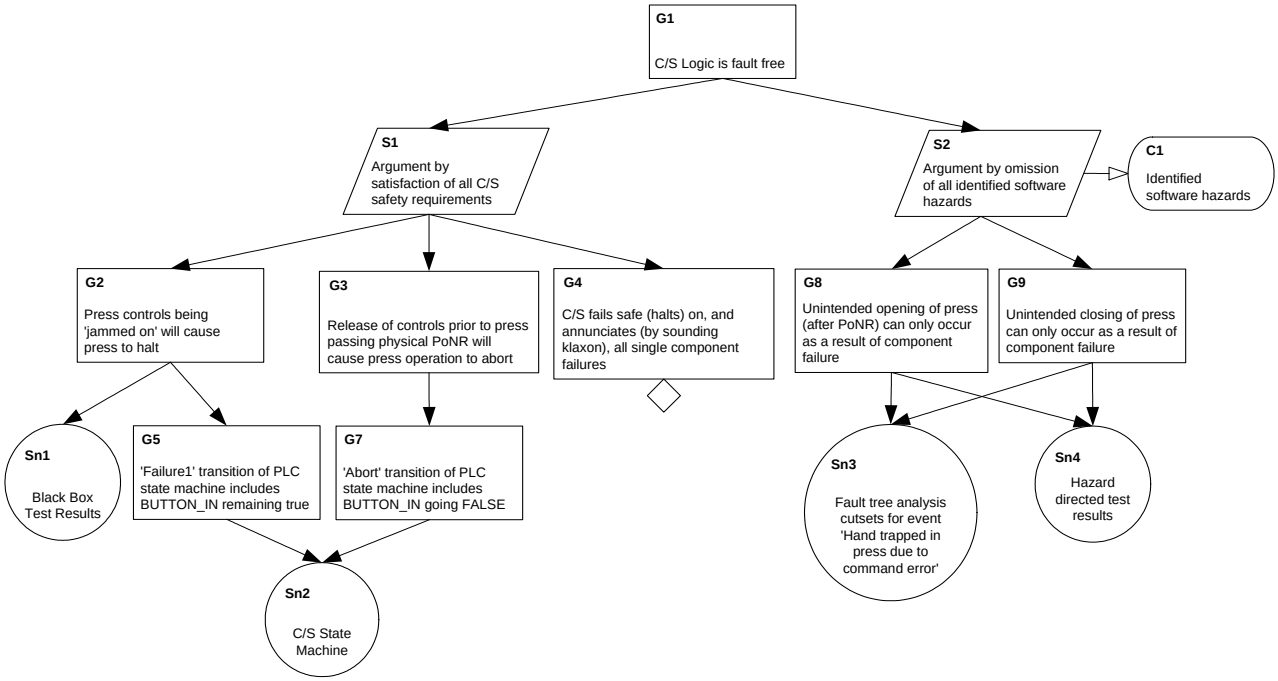

Figure 3 – An example safety argument goal structure

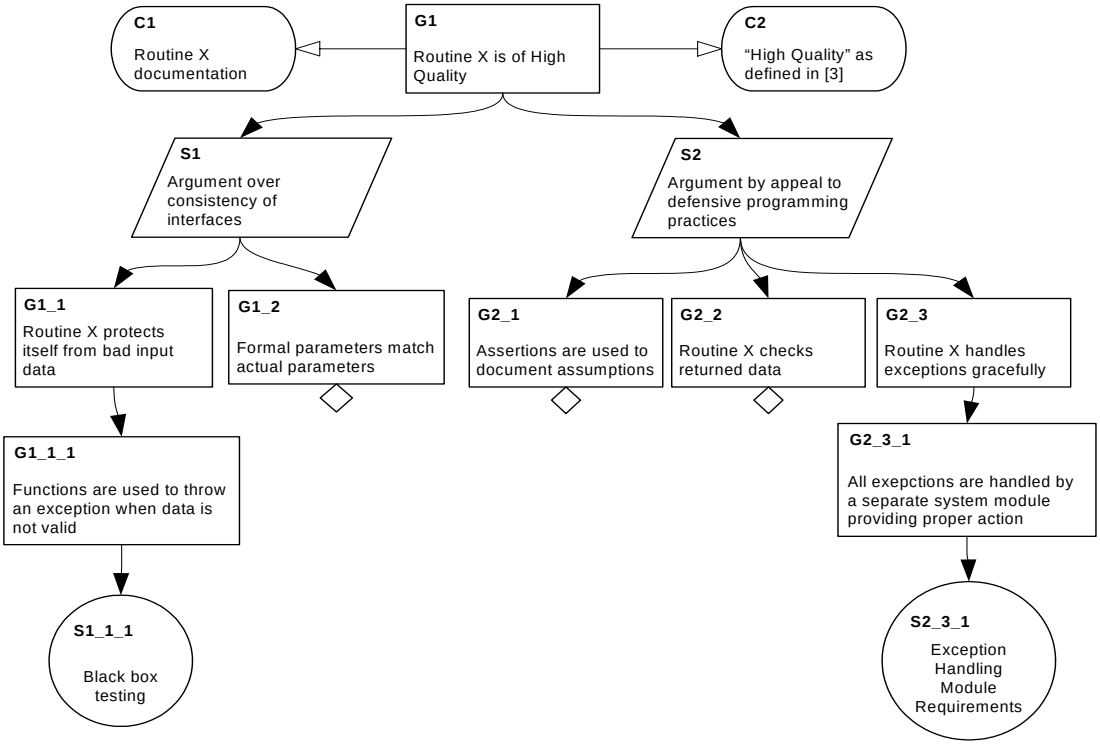

Figure 4 – An example software quality goal structure

development process, such that the precise role each development activity takes in satisfying specific requirements can be identified.

The lightweight, yet structured, approach of GSN to determining what evidence is required to satisfy requirements means that the notation should be applicable to a variety of evidence-based software engineering approaches. While the GSN arguments used in safety-critical industries are often formally assessed as part of a certification process, many of the benefits of GSN come from its use as a notation to identify specific low-level requirements for evidence and demonstrate how different types of evidence are combined to meet requirements. GSN arguments can be used as part of discussion exercise which explicitly identifies the role each item of evidence plays within a larger set of evidence. Breaking down high-level requirements and objectives into low-level requirements which can be satisfied by individual items of evidence, clarifies the link between requirements and evidence.

It is feasible for arguments to be developed using the notation about many properties of software including reliability, quality and trustworthiness. Arguments could also be constructed to address claims regarding the efficacy of software development or assessment processes. Through argument construction, a dialogue can be generated about the suitability of different argument approaches and evidence in demonstrating the satisfaction of claims regarding product *or* process attributes. Depending upon what attributes or properties needed to be demonstrated about the software or the software development process, an argument decomposition process would be followed which would start from the high level requirements and decompose the argument until individual items of evidence could be identified which satisfy the low-level goals.

The use of GSN within evidence-based software engineering would follow a similar process to its use in the development of arguments for safety critical systems. Identification of high level requirements for the software product or process usually occurs early on in the system/software development process. These requirements could be about any attribute of the product/process which the developers wished to demonstrate through the use of evidence. These requirements would be broken down into lower level claims, the set of which usually relates to a combination of both product and process evidence. This decomposition would be described in the argument. Performing the decomposition at the beginning of the software development process allows specific items of evidence required for validation of the high level requirements to be identified early on. The argument structure and supporting evidence guides the software development process identifying the evidence about the product and process which must be generated. It does not specifically provide a software development process timeline, however many existing techniques for the management of software engineering processes can be used to do this. Instead the goal structure concentrates on *why* particular evidence is being generated.

A simple (and incomplete) example is given in Figure 4, which shows how an argument structure could be developed for demonstrating using evidence that a particular routine (Routine X) is of high quality. This argument structure is based upon requirements identified in [3] for a high quality routine. The argument is based upon two strategies for demonstrating "high

Table 1 – Objectives, Arguments and Evidence for the Dependability Attributes

	Objective	Typical Argument	Typical Evidence
Safety	System is adequately safe	Hazard mitigation argument	Hazard Analysis, Causal analysis
Reliability	System meets reliability requirements	Enough redundancy, resilient components	Testing / Simulation, Markov analysis
Maintainability	System meets maintainability requirements	Modular cohesive design, plug and play devices, ease of replacing components	Expert opinion, simulation.
Security	Mission critical information is adequately protected	Assets protection argument	Access control, policies, MOATs

quality": consistency of interfaces and defensive programming. It is not claimed in this paper that these strategies are the most appropriate for showing that a routine is of high quality (a strategy for demonstrating such a property would need to be determined for a particular piece of software). However, in the example we can see how the top claim (G1) is broken down into claims about consistency of interfaces (G1_1 & G1_2) and claims about defensive programming (G2_1, G2_2 & G2_3). Goal G1_1_1, which provides support for Goal G1_1, is shown to be true by appeal to black box testing evidence, while Goal G2_3_1, which provides support for Goal G2_3, is shown to be true by appeal to evidence in the form of Exception Handling Module Requirements. Further argument and evidence would need to be developed for the undeveloped goals (G1_2, G2_1, G2_2). In the example we can see how requirements for evidence and items of evidence are identified, and thus how the notation can be used as a basis for evidence-based software engineering.

4.1 Patterns of Argument

Where there is consensus in a community about a suitable argument approach, agreement can be demonstrated through the development of an argument pattern. Common structures in safety case arguments are reused through their documentation as 'Safety Case Patterns'. Arguments patterns provide descriptions of generic argument fragments, which are reuseable in a variety of different arguments. In [4] a description of a safety case pattern language based on the Goal Structuring Notation is presented. Similarly, Safety Case AntiPatterns [2 & 5] can be used to communicate weak and flawed safety arguments, such that they may be recognised and avoided in future developments.

With respect to software engineering, argument patterns for combining evidence could be developed to record generic approaches for meeting requirements. Where there is consensus in the software engineering community about how certain items of evidence can be combined to demonstrate a property of software, this consensus can be captured in a pattern. The results of large studies, which combine evidence from various software developments can be recorded in the form of patterns. These patterns would describe how certain combinations of evidence can be used in software development to demonstrate certain properties.

4.2 Modular Arguments

In order to manage complex *safety* cases – in which there are complex relationships between arguments – the principles of compositional, modular, safety cases have already been established [6]. In this approach, the safety case for an overall system can be divided into a number of modules – containing the separate arguments and evidence for different aspects of system safety. For example, with a complex avionics platform the overall safety case can be reasoned about as the composition of separate arguments for each of the separate avionics subsystems. These separate arguments often correspond to components of the system architecture.

However, these individual modular arguments cannot be reasoned about in isolation. For example, it may only be possible to argue about the safety of one avionics subsystem in the context of assumed safe behaviour of another. To help manage the relationships that exist between safety case modules the concept of modular safety case interfaces (defining clearly the objectives, evidence, and assumed context of the case together with any dependencies on other cases) and safety case contracts (recording how the dependencies between safety cases are resolved) have been defined. It is these principles of modularising an overall argument into separate but interrelated arguments that we believe offers a way forward in representing arguments and evidence for dependability.

5. DEPENDABILITY ARGUMENTS

In addition to safety, there are also other important attributes that need to be addressed as part of software engineering, such as maintainability and security. We use the term Dependability as a system property that is perceived to encompass these wider attributes. Shifting the focus from only one attribute (i.e. safety) and attempting to address all attributes equally, poses two main obstacles that need to be overcome; the evolution of this type of argument, and the resolution of conflicts between evolving (into specification) attribute objectives.

Although the dependability case is a relatively new and untested concept, the idea of developing 'cases' for system attributes other than safety is not new. Reliability and Maintainability cases are, for example, a requirement of the U.K. Defence Standard 00-40 [7]. Similarly, the Information Technology Security Evaluation Criteria (ITSEC) [8] require an explanation of why a system has met its required security level – implicitly this is a demand for a 'security case'.

Table 1 shows typical objectives, arguments and evidence required when reasoning about different dependability attributes. The main issues when developing arguments for multiple dependability attributes is their non-orthogonal nature and the fact

that they can be interrelated to each other. Attempting to address all dependability attributes can result in competing objectives. For example, it is a common problem for availability to be in conflict with safety (e.g. in civil aviation). As a consequence, there are trade-offs among the dependability attributes that need to be resolved in order to achieve the optimum dependability characteristics for the system. Conflict may arise explicitly during specification stage (i.e. two directly conflicting requirements), or implicitly during design (i.e. a chosen design achieves one attribute's criteria but compromises another). When a conflict is encountered during the development of the system/argument then the developers need to make a trade-off – representing the best possible compromise between the conflicting objectives. The different attributes are heterogeneous and have different measures (such as safety risk, security risk and mean time to failure MTTF) and make use of different techniques (as illustrated in Table 1). As a consequence it is very difficult a common basis of comparison.

As an illustration, Figure 5 presents an abstract overview of the dependability case architecture viewed in terms of separate modules of argument and evidence. Although separate arguments are presented for individual dependability attributes, each of the arguments is clearly set in the context of a trade-off argument that will separately argue about the resolution of trade-offs between conflicting dependability objectives. In addition to relating dependability arguments by means of the trade-off argument it is also possible that there may be more direct relationships that need to be captured between the arguments where they directly support one another.

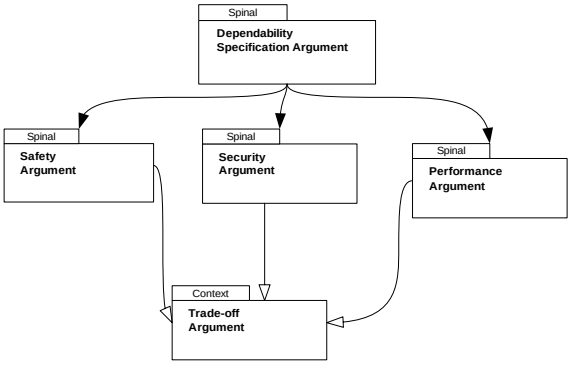

Figure 5: A modular view of a dependability argument

For example, a reliability argument may support a safety argument or there may be arguments and evidence that can be used to support both the safety *and* security argument. Such relationships have not been illustrated in the figure.

Modular arguments have the potential to represent a multi-attribute dependability argument and provide the basis for a dependability case that explicitly represents and reasons about the relationship between attributes. Exploring this potential is an objective of our current research. We believe that this modular approach is applicable to combining evidence for different properties of software to meet the variety of requirements that exist in software engineering.

6. ASSURANCE OF ARGUMENTS

Implicit in the assessment of a safety case is a consideration of whether the safety argument has been *sufficiently* assured with the evidence available. In this section we present ongoing research into an approach for considering and explicitly describing argument assurance [5]. The approach described is a process which occurs in the development and assessment of a GSN argument but currently remains unexpressed.

The argument claims made in GSN goals are propositions. These propositions can be qualitative or quantitative and may be subjective in nature. However, the statements are either true or false. For example, the statement "failure rate of component X is 10^{-4} failures per operational hour" is either a true or false. This characteristic of statements leads to arguments having properties based upon the truth or falsity of the statements. In argumentation, the strongest arguments are designed to be both Valid – if premises are true, conclusion is true – and Sound – an argument which is valid and has true premises.

It is desirable to develop safety arguments that are both valid and sound. However, due to the evidence typically available and the inferences that must be made, a provably valid and sound argument is unobtainable for a most systems. Thus, GSN accepts arguments that are consistent – if premises are true, conclusion may be true – and thus causally weaker. This weaker form of causal relationship is known as *inductive* argumentation – the conclusion follows from the premises not with necessity but only with probability. While the stronger, valid argument form is known as *deductive* argumentation – if premises are true, then the conclusion must also be true.

The inductive nature of GSN arguments implies that a level of probability must be associated with the satisfaction of an argument. It is not the case for goal structures that the top-level goal is true because all of the solutions (evidence) are true. Instead, the aim of the argument is to show the sufficiency of the child goals and solutions in satisfying the parent goal. While GSN describes the relationship between premises and conclusions, currently it does not capture the inductive nature of the argument.

For inductive arguments it can be useful to express the relevance of each child element in satisfying the parent goal and the strength of the argument step as a whole. It is beneficial to make explicit the connectivity within the causal relationships between parent goals and child goals/solutions. This will clarify the sufficiency of the premises (solutions) in satisfying the conclusion (top-level goal). By making explicit the *strength* of the argument the knowledge captured within the goal structure will be increased. Thus the argument is both improved and made more transparent.

The term *Assurance* inherently expresses the subjectivity when determining the strength of an inference. It also encapsulates the concept of confidence, which is part of the objective of a safety argument – the determination of the confidence that can be placed in the safety of a system. Assurance is a property of an argument's conclusion. It is based upon the likelihood that the premises are true (i.e. the assurance of the premises) and the extent to which the premises entail[1] the conclusion. The overall

[1] To involve, logically necessitate (a particular conclusion)

assurance of a safety argument is equal to the assurance of the top-level goal of that argument. We define Assurance as: *A qualitative statement expressing the degree of confidence that a claim is true.*

The size and complexity of arguments about systems, combined with the subjective nature of argument composition is such that assurance cannot easily be considered quantitatively. Instead, we believe a qualitative approach, expressing levels of assurance, similarly enables articulation of the strength of arguments. Assessment of assurance can be a qualitative judgement based upon an understanding of the child element to parent goal inference. Through expressing assurance, these judgements are made explicit within the argument. In the following two sub-sections two particular issues relating to argument assurance are discussed.

6.1 Argument Decomposition – Support Type

When developing an inductive argument, there is a requirement for the argument to be cogent (i.e. the premises give good rational support to the conclusion). Within GSN, the assurance assigned to the parent goal determines the required cogency of the argument step from child to parent goal. Using GSN, arguments are typically constructed in a top-down fashion such that suitable premises are developed which show the acceptability of the conclusion. Child goals and/or evidence that support the goal are identified and these child elements must be suitable to produce a cogent argument. The level of support is determined by the assurance of the child elements and the extent to which the child elements entail the conclusion or Parent Goal. In assurance decomposition, the minimum required assurance level of each child element is determined. The assurance decomposition process determines the level of assurance that is required of the child elements in order to sufficiently assure the parent goal. Once the goal structure is completed confirmation can be provided (bottom up) that the evidence referenced sufficiently assures the basic premises of the argument and that those premises ultimately sufficiently assure the conclusion of the argument. In this way confirmation can be provided that the argument is cogent and the top-level goal has been sufficiently satisfied.

The first stage to determining argument sufficiency is to establish the relevance of each individual child element to the parent goal - *the extent to which the child element entails the parent goal.* A child element on its own can either totally satisfy the entire parent goal or can partially satisfy the parent goal. Argument support provided by the child element set can have one of three forms. Govier, in [9] identifies these three types of argument support (Figure 6).

A child element that satisfies the entire parent goal provides single support. The relevance of a child element is determined, by considering whether the truth of that child element statement entails the truth of the parent goal statement. Highly relevant child elements which each fit a single support argument can be used in combination to produce a convergent support argument. A child element that does not fit a single support argument may, in combination with other child element(s), fit a linked support argument. With a linked support argument each child element addresses a different aspect of the parent goal. Each child goal is required and the argument could not be supported if any of the child elements were removed. With a convergent support argument each child element independently addresses the whole of the parent goal. Child elements that fit a convergent argument are identified by assessing the relevance of the child element. Child elements that fit a linked support pattern are identified by determining whether the element is required.

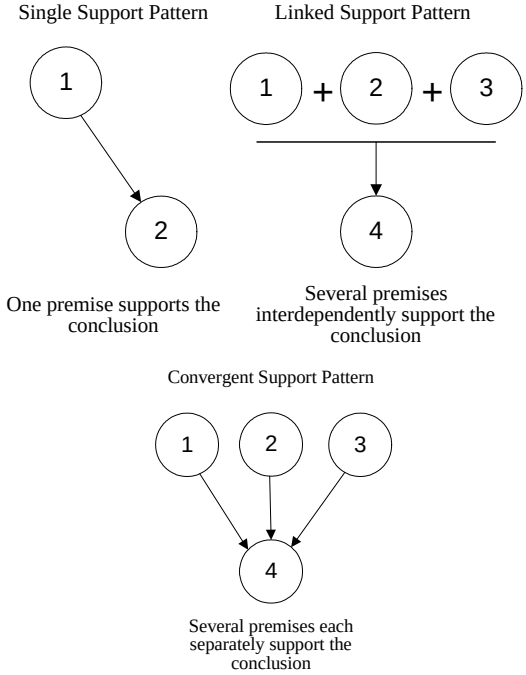

Figure 5: Govier's support pattern types [9]

Within an argument step it is beneficial that all child elements fit a linked support argument, or all child elements fit a convergent support argument. This aids the production of a clear and understandable argument and helps to determine the required assurance of the child elements. A hybrid argument structure that is neither linked nor convergent can appear ambiguous in how it satisfies the parent goal. It can be difficult to discriminate between distinct threads of argument and those that reinforce one another.

6.2 Determining Evidence Assurance

An important stage in assurance apportionment is determination of the assurance of an item of evidence. After determining the Top-level assurance level and decomposing this throughout the argument structure, the solutions will have a required level of assurance that must be met. The required assurance of an item of evidence is determined in the same way as the assurance of a child goal is determined (i.e. by considering the support type, relevance and independence). As well as determining this required assurance for a solution it is necessary to demonstrate that this level of assurance has been achieved.

The assurance level of a solution is an expression of the process evidence related to generating the evidence. This identifies the trustworthiness of the evidence and is thus based upon a number of factors. These factors include, but are not limited to:

- "Buggy-ness" – how many "faults" there are in the evidence presented;
- Level of review;

- For tool-derived evidence: Tool Qualification and Assurance;
- Experience and Competence of the personnel.

There are certain factors that will invalidate the evidence (and thus give it a assurance level of zero). Examples of this include applying the evidence to the wrong version of the item under test and faults or limitations of the tool/technique which undermine the evidence.

The factors identified above will apply to most types of evidence, however other factors which should be taken into consideration, when determining the assurance of the evidence, will vary depending upon the type of evidence. Assessment of the assurance of individual items of evidence is an ongoing area of research for us.

7. CONCLUSIONS

The Goal Structuring Notation is a well established technique use in safety critical industries for identifying requirements for evidence, documenting how evidence can be combined and assessing large sets of evidence. Ongoing research is considering how evidence can be weighed through the consideration of assurance and how arguments for other dependability properties can be constructed with necessary trade-offs made. We believe that the notation provides a powerful technique which can be used in the field of Software Engineering to help realise evidence-based software development.

8. REFERENCES

[1] U.K. Ministry of Defence, *00-56 Safety Management Requirements for Defence Systems*, Ministry of Defence, Defence Standard, December 2004.

[2] T. P. Kelly, *Arguing Safety – A Systematic Approach to Safety Case Management*, DPhil Thesis YCST99-05, Department of Computer Science, University of York, UK, 1998.

[3] S. McConnell, *Code Complete– A Practical Handbook of Software Construction*, Microsoft Press, 1993.

[4] T P Kelly & J A McDermid, "Safety Case Construction and Reuse using Patterns", in *Proceedings of 16th International Conference on Computer Safety, Reliability and Security* (SAFECOMP'97), Springer-Verlag, September 1997.

[5] R. A. Weaver, *The Safety of Software – Constructing and Assuring Arguments*, DPhil Thesis, Department of Computer Science, University of York, UK, 2003.

[6] I J Bate & T P Kelly, "Architectural Considerations in the Certification of Modular Systems", in *Proceedings of the 21st International Conference on Computer Safety, Reliability and Security* (SAFECOMP'02), Springer-Verlag, September 2002.

[7] U.K. Ministry of Defence, *00-40 - Reliability and Maintainability (R&M)*, June 2003.

[8] European Community advisory group Seniors Ocals Group Information Systems Security. *Information Technology Security Evaluation Criteria*. Department of Trade and Industry, United Kingdom, June 1991. Version 1.2.

[9] T. Govier, *A Practical Study of Argument.* Wadsworth, 1998