Safety Case Composition Using Contracts - Refinements based on Feedback from an Industrial Case Study

Jane Fenn and Richard Hawkins
BAE SYSTEMS, Brough, UK
Phil Williams
General Dynamics (United Kingdom) Ltd, Hastings, UK
(representing the Industrial Avionics Working Group)

Tim Kelly
University of York, York, UK

Abstract

Modular safety cases provide a means of organising large and/or complex safety cases into separate but interrelated component modules of argument and evidence. Safety case 'contracts' can be used to record the interdependencies that exist between safety case modules – e.g. to show how the claims of one module support the arguments of another. A number of techniques for structuring and describing modular safety cases using the Goal Structuring Notation were defined by Kelly in (Kelly 2001). The Industrial Avionics Working Group, (IAWG) has been using these techniques as part of a substantial industrial case study being funded by the UK Ministry of Defence. Based on this experience, and a number of issues encountered, modifications to the original approach have been defined. This paper presents some of these experiences of the IAWG in using 'modular' GSN – in particular, those relating to capturing and recording safety case contracts – and proposes an enhanced approach.

1 Introduction

The Industrial Avionics Working Group, (IAWG), which was formed in 1979, is an industrial consortium of companies working in the aerospace sector, namely, BAE SYSTEMS, General Dynamics (United Kingdom) Ltd, Westland Helicopters, Smiths Aerospace and SELEX S&AS. During 2006, the Ministry of Defence has funded a programme of research, building on a feasibility study carried out by IAWG, and developing a modular safety argument for an aircraft system. This has entailed the use of the modular Goal Structuring Notation (GSN) extensions defined by Kelly (2001). This activity has highlighted some issues for which IAWG, in conjunction with Kelly, have proposed some modification and enhancements to the definition of the modular GSN extensions, with accompanying guidance on implementation issues.

2 Modular GSN Definition

GSN has been widely adopted by safety-critical industries for the presentation of safety arguments within safety cases. However, to date GSN has largely been used for arguments that can be defined 'stand-alone' as a single artefact rather than as a series of modularised interconnected arguments. In order to make the GSN support the concepts of modular safety case construction it has been necessary to make a number of extensions to the core notation.

The first extension to GSN is an explicit representation of modules themselves. This is required, for example, in order to be able to represent a module as providing the solution for a goal. For this purpose, the package notation from the Unified Modelling Language (UML) standard has been adopted. The GSN symbol for a safety case module is shown in Figure 1.

In presenting a modularised argument it is necessary to be able to refer to goals (claims) defined within other modules. Figure 1 introduces an element to the GSN for this purpose – the "Away Goal". An away goal is a goal that is not defined (and supported) within the module where it is presented but is instead defined (and supported) in another module. The Module Identifier (shown at the bottom of the away goal next to the module symbol) should show the unique reference to the module where the goal can be found.

Away goals can be used to provide ***support*** for the argument within a module, e.g. supporting a goal or supporting an argument strategy. Away goals can also be used to provide contextual backing for goals, strategies and solutions.

Representation of away goals and modules within a safety argument is illustrated within Figure 1. The annotation of the top goal within this figure "SysAccSafe" with a module icon in the top right corner of the goal box denotes that this is a 'public' goal that would be visible as part of the published interface for the entire argument shown in 1 as one of the "objectives addressed".

The use of some of these notational extensions by the IAWG in developing the modular safety argument has highlighted issues which are discussed in Section 3.

The strategy presented within Figure 1 to address the top goal "SysAccSafe" is to argue the safety of each individual safety-related function in turn, as shown in the decomposed goals "FnASafe", "FnBSafe" and "FnCSafe". Underlying the viability of this strategy is the assumed claim that all the system functions are independent. However, this argument is not expanded within this "module" of argument. Instead, the strategy makes reference to this claim being addressed within another module called "IndependenceArg" – as shown at the bottom of the away goal symbol. The claim "FnASafe" is similarly not expanded within this module of argument. Instead, the structure shows the goal being supported by another argument module called "FnAArgument", indicated by the 'module reference' symbol. The "FnBSafe" claim is similarly shown to be supported by means of an Away Goal reference to the "FnBArgument" module. The final claim, "FnCSafe", remains undeveloped (and therefore requiring support) – as denoted by the diamond attached to the bottom of the goal.

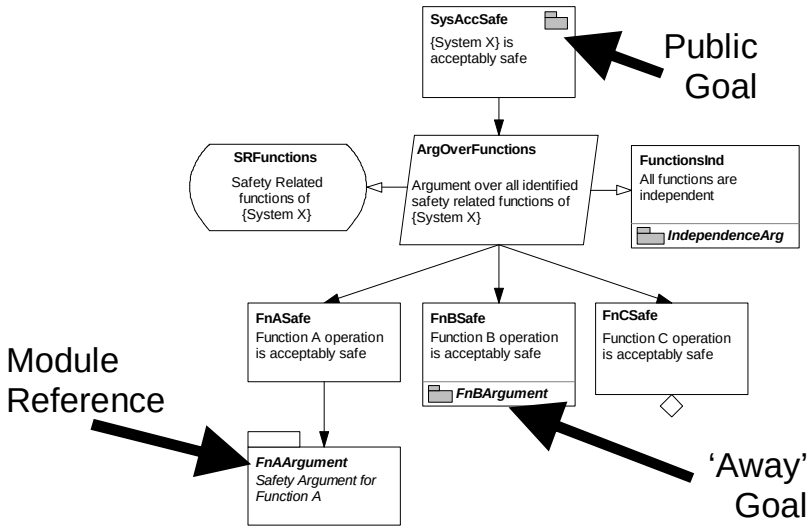

Figure 1 GSN Extension

In the same way that it can be useful to represent the aggregated dependencies between software modules in order to gain an appreciation of how modules interrelate 'in-the-large' (e.g. as described in the 'Module View' of Software Architecture proposed by Hofmeister et al. in (Hofmeister et al. 1999) it can also be useful to express a module view between safety case modules.

If the argument presented within Figure 1 was packaged as the "TopLevelArg" Module, Figure 2 represents the module view that can be used to summarise the dependencies that exist between modules. Because the "FnAArgument" and "FnBArgument" modules are used to support claims within the "TopLevelArg" module a supporting role is communicated. Because the "IndependenceArg" module supports a claim assumed as context to the arguments presented in "TopLevelArg" a contextual link between these modules is shown.

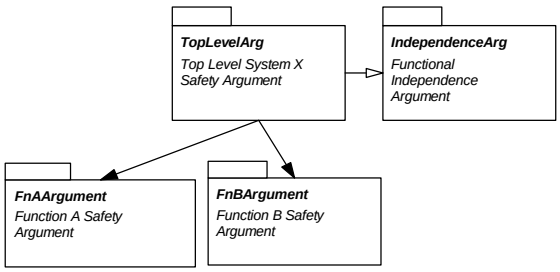

Figure 2 – Example Safety Argument Module View

In a safety case module view, such as that illustrated in Figure 2, it is important to recognise that the presence of SolvedBy relationship between the module TopLevelArg and FnAArgument implies that there exists at least one goal within TopLevelArg that is supported by one or more arguments within FnAArgument. Similarly, the existence of an InContextOf relationship between TopLevelArg and IndependenceArg implies that there exists at least one contextual reference within TopLevelArg to one or more elements of the argument within IndependenceArg.

Alongside the extensions to the graphical notation of GSN, the following supporting documentation is required:

Interface declaration for each safety case module – the external visible properties of any safety case module must be recorded – e.g. the goals it supports, the evidence (solutions) it presents, the cross-references ('Away Goal' references) made to / dependencies upon other modules of argument. Figure 3 depicts the items to be defined on the boundary of a safety case module expressed using the GSN.

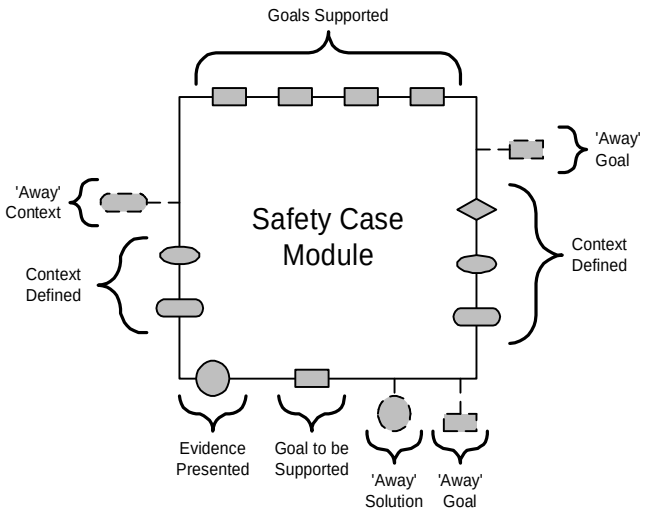

Figure 3 – The Published Interface of a GSN Safety

The overall safety case is composed from the safety case modules by linking elements in different modules in a 'safety case contract', such as goals requiring support from one safety case module are solved by public goals in a second safety case module. Kelly proposes a table is used to record which elements are provided or resolved by the contract and context which is consistent between the modules. It is use of these tables to record safety case module contracts that IAWG found to be difficult in practice and so have proposed an alternative strategy, as described in section 4.

3 Issues of Using Modular GSN Notation

Initial concerns arose when using the 'Module Reference' notation within a safety case module. An example below represents where the computing architecture provides functions that prevent applications running on it from communicating other than by pre-defined mechanisms. The safety case module discussing the need to prevent unintended communication between applications doesn't need to know how the architecture provides that capability, but does need to know that the architecture safety case module will provide that argument, so the following GSN fragment represents this situation using the 'module reference' symbol.

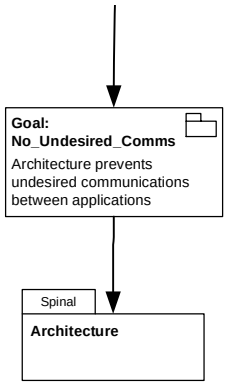

Figure 4 - Safety Argument Fragment in the Application Safety Case Module

This 'module reference' symbol had been used as a way of indicating that the goal would be solved using some goal (or goals) in the named module, (with the link explicitly defined in a separate safety case contract). This is distinct from an 'away goal', which references a specific goal in another module, such that it is essentially *hard-wired*, thus not requiring a separate safety case contract. In the example in Figure 4, the claim 'Goal: No_Undesired_Comms' is to be solved using a goal (or goals) contained within the Architecture module. The specific goal (or goals) from the Architecture safety case module that are to be used would be specified in the safety case contract.

Where the module containing support for a goal is not known in advance, Kelly proposes the use of the 'undeveloped' annotation. Using this approach in the example in figure 4, the goal requiring support, 'Goal: No_Undesired_Comms' would simply have been left as an undeveloped goal and the module reference element would not have been used.

Neither of the two approaches to representing a goal requiring support discussed above was found to be ideal when applied to the case study. Below, the issues and challenges are highlighted.

3.1 Undeveloped goal approach

The use of the 'undeveloped goal' notation where a goal is supported by argument in another module raised some concerns:

- It is not possible to distinguish between goals requiring support from other modules (i.e. those requiring a safety case contract), and those that require further development, i.e. the method of development is unclear
- Even once the contract is in place, there is no way of identifying where the contract is made, or the modules that are linked, as the goal remains represented as an undeveloped goal, i.e the GSN provides no visibility of the contractual inter-module links

These drawbacks are balanced by providing a representation which does not 'hard-wire' the argument into any safety case architecture constraints, i.e. changing the supporting argument module does not require the calling safety case module to change, only the safety case contract linking the two modules needs to change.

3.2 Module reference approach

The module reference approach provides greater visibility that a goal is supported by another safety case module, but the argument becomes 'hard-wired' to an extent. The argument module developer is forced to identify up-front the module that is going to provide support for the goal. This doesn't permit the desired flexibility to allow changes to the way a goal is solved using other modules; it also puts the requirement upon the developer of the module to identify the way in which the goal will be discharged by other modules.

3.3 Summary of GSN Notational Issues

Clearly a 'trade-off' has been identified between the visibility of the definition of links between safety case modules and maximising the reusability and modifiability of modules by minimising the 'hard-wiring' between safety case modules.

4 Issues of Using Safety Case Contract Tables

Kelly (2001) describes the use of Safety Case Contracts as a matching between 'goals requiring support' (expressed as undeveloped goals or module references) and 'goals providing support' (expressed as public goals) across safety case module boundaries. Defining the goals that are public (and hence available to provide support to other modules) and those that are private (and not available to provide support) has raised further issues, discussed below.

4.1 Public and Private Goals

Once the goals requiring support from other modules have been identified, it is necessary to record the goals defined in other modules that are to be used to provide this support, by means of a safety case contract. These goals providing support are often referred to as the *public goals* of the safety case module. Kelly (2001) notes that the interface should not necessarily contain all of the goals supported by the module, owing to the fact that some will be considered internal detail whilst others will not.

It is possible for any goal to be declared public, but this may not necessarily be desirable, particularly if modules are being developed independently, in which case it would need to be negotiated explicitly as to which goals are required to be public. It is desirable that the number of public goals should be as restricted as possible. Using only the minimum necessary public goals eases assessment of the impact of changes on other modules. There is however a trade-off between easier assessment of change (which requires a small number of public goals) and reusability (which is easier when more goals have been declared public).

In order to develop a modular safety case, the argument integrator may need visibility of private goals in modules, and then request the goal 'owner' to make the required goals public in that module. It should not be possible for anyone other than the 'owner' of the module to change the public/private status of a goal. It may be useful to try to enforce this through tool support.

If goals which have been declared public are not used to discharge a goal requiring support from another module in a given safety case architecture configuration, then it should be made clear that this is the case, as they are not then of concern when considering the impact of changes on other modules. Therefore it may be necessary to indicate in some way which public goals are unused for a particular safety case, such that it is clear that whilst the goals are 'visible' to other modules, they are not required.

4.2 Capturing Safety Case Contracts

Whenever a successful match can be made between goals requiring support in one module, and goals provided in another module, a contract is made to capture the agreed relationship between the modules. Kelly (2001) proposes a table to be used for capturing the contractual relationship as shown in Table 1.

Safety Case Module Contract			
Participant Modules			
(e.g. Module A, Module B and Module C)			
Goals Matched Between Participant Modules			
Goal	*Required by*	*Addressed by*	*Goal*
(e.g. Goal G1)	(e.g. Module A)	(e.g. Module B)	(e.g. Goal G2)
Collective Context and Evidence of Participant Modules held to be consistent			
Context		*Evidence*	
(e.g. Context C9, Assumption A2)		(e.g. Solutions Sn3, Sn8)	
Resolved Away Goal, context and Solution References between Participant Modules			
Cross Referenced Item	*Source Module*	*Sink Module*	
(e.g. away Goal AG3)	(e.g. Module B)	(e.g. Module C)	

Table 1 - Safety Case Contract Table

In trying to apply this tabular approach to an example case study modular safety case a number of problems were encountered, including:

- It was unclear without more explicit examples, exactly what the safety case contract table was meant to cover, and how it was to be applied. In practice it was found to be difficult to capture all the necessary information in such a tabular form.
- There is no mechanism for capturing the strategy used in addressing one goal with another. This strategy could in many cases be fairly complex. In the same way that strategy (potentially with its own context and assumptions) may be needed to show how a goal *within* a module solves another, this may also be required where the solution is made across modules via the contract.
- The tables exist as completely separate entities from the GSN argument itself. This means that there is no visibility within the GSN structure of contractual links.

To address these concerns, the IAWG team have proposed an alternate approach to capturing safety case contracts. This approach is currently being trialled on an industrial case study.

5 IAWG Proposed Implementation of Safety Case Contracts

Based on the challenges identified above, the following solution has been proposed as a way of capturing the safety case contracts between safety case modules in the IAWG case study modular safety argument.

IAWG propose that the contract should be captured using GSN, as this provides an expressiveness and clarity which is not provided by the use of a tabular approach. This also allows the contract to be integrated with, and viewed as part of, the total safety case argument.

5.1 GSN Contract Reference

The contract will be constructed as a GSN safety case module which can be referenced by the goal requiring support. This means that the module and goal providing the solution to the goal requiring support is not identified directly by that goal, but is instead specified in the GSN contract module. This allows the solution in the contract to be changed without the module containing the goal requiring support being changed. Figure 5 illustrates the notation that is proposed to indicate that a goal is to be solved using a goal or goals provided by other modules, using a safety case contract.

In the example shown in Figure 5, the goal ‘Goal: No_Undesired_Comms’ is to be solved via the safety argument contract ‘Contract {Z}’. It can be seen that a new GSN symbol has been introduced to represent the contract module. This new symbol has been introduced here specifically to distinguish a safety case contract module from a ‘normal’ safety argument module; this is necessary as there are certain

properties of safety case contract modules which do not apply to safety argument in general. These unique properties of safety case contract modules are discussed later.

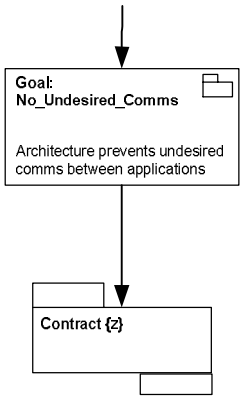

Figure 5 – Safety Case Contract Reference in GSN

It may be desirable to express the fact that a goal requiring support will be solved through use of a contract without specifically making reference to a particular safety case contract module. This may be desirable if, for example, the solution to the goal has not yet been defined in a particular contract module. In such a situation, the intention to support a goal requiring support through use of a contract can be indicated through using the GSN goal annotation proposed in Figure 6.

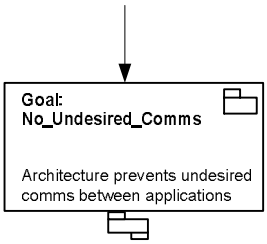

Figure 6 - Indicating a Goal is to be solved using a contract

Once a contract is developed to provide a solution to the goal, the contract can be referenced explicitly as in Figure 5.

5.2 GSN Contract Module

The contract itself is represented as a GSN module. This shows how the goal requiring support from one module is solved using a goal, or goals, provided by other modules. An example ‘Contract {Z}’ module is shown in Figure 7.

This contract shows how the unresolved goal ‘Goal: No_Undesired_Comms’ from the Applications module (identified using an away goal reference) is resolved using a goal ‘Goal: Partitioning’ from the Architecture module. A highly simplified version of the Architecture module, provided for illustrative purposes, is shown in

Figure 8. This goal is similarly identified using an away goal to ‘Goal: Partitioning’ in the Architecture module. A strategy is also provided.

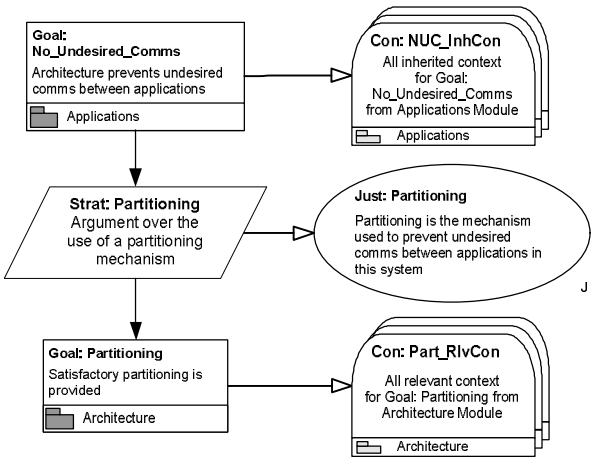

Figure 7 - Contract {Z}

It can be seen in Figure 7 that the context relevant to each away goal in the contract must also be included. We discuss later the issue of identifying relevant context. Another new GSN symbol is required at this point in order to indicate that the context on the goal is a ***collection*** of existing contexts (in this case from other modules). The ***away*** context ‘collection’ symbol is illustrated in Figure 7. It should be noted that this symbol is equivalent to including many ‘away context’ references to each element of existing context in the other module, however the new ‘context collection’ symbol allows the presentation to remain less cluttered. The contextual elements that are covered by the ‘context collection’ must be stated.

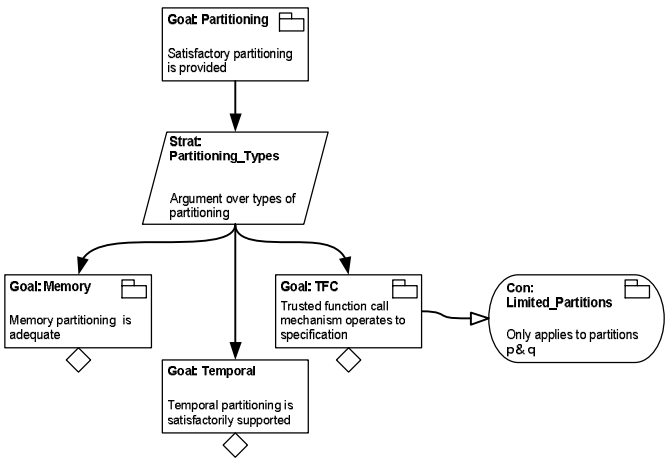

Figure 8 - Architecture Safety Case Module (simplified)

The context relevant to ‘Goal: No_Undesired_Comms’ in Figure 7 is not only context directly connected to this goal in the Applications safety case module, but also ‘inherited’ context from any parent goals higher up the argument structure. Similarly, the context for goal ‘Goal: Partitioning’ must include not only all inherited context and context directly connected to the goal but also must take into account where lower level goals are reduced in scope by the use of contexts, assumptions or justifications. The reason for this is illustrated in Figure 8 where the context ‘Con; Limited_Partitions' reduces the applicability of the solution offered from all partitions to only partitions p and q. It is therefore necessary for ‘Con; Limited_Partitions' to be included as part of the collective context to ‘Goal: Partitioning’ in the Contract {z} module.

A justification is provided in the contract module through ‘Just: Partitioning’ which justifies why ‘Goal: No_Undesired_Comms’ is supported by ‘Goal: Partitioning’ within the scope defined by the inherited context of ‘Goal: No_Undesired_Comms’. As seen, it is also possible to include a strategy between the goals matched in a contract module, if this is required.

5.3 Generic Pattern for GSN Safety Argument Contracts

The specific example above has been used to illustrate how a GSN safety argument contract module approach may be applied. It is possible to define a contract module in more generic terms as a pattern which can be used in constructing a contract for any goal requiring support from other modules. A contract pattern is proposed in Figure 9.

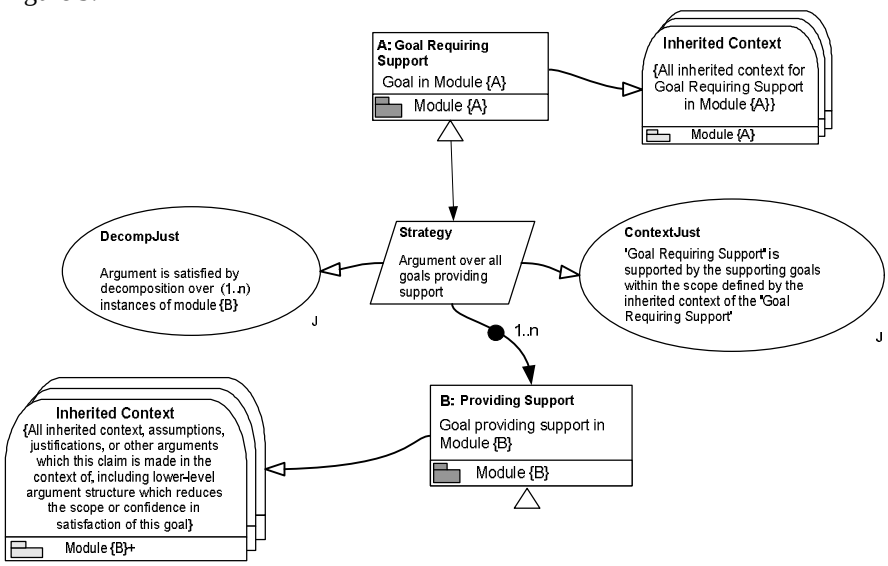

Figure 9 - Generic pattern for safety case contract modules

Figure 9 illustrates how more than one goal from more than one module may, if required, be used to resolve the top goal. Strategy and justification elements may be used as necessary to make the argument clear.

5.4 Dealing with Context

When dealing with context in making safety argument contracts between modules, Kelly (2001) talks about agreeing the consistency between the collective context of the participating modules. In practice, this can be extremely problematic. The simplest way of showing consistency between collective context is if there is a direct match between the contexts. In reality this is never, as Kelly asserts, likely to be the case. It is unrealistic to expect that context which is inherited from modules which have been developed independently and in significantly different domains might match. Anything other than such a direct match is likely to make compatibility extremely complex to argue. For example, consider contract {Z} in Figure 7. It is possible that context defined in the Architecture module which is inherited by goal 'Goal: Partitioning' may, for example, refer to modes of the operating system. Such information is unlikely to appear as context in the Applications module, as this module may have no knowledge of the modes of the operating system. Although the context of 'Goal: No_Undesired_Comms' and 'Goal: Partitioning' would, in this case not directly 'match', it doesn't necessarily mean that 'Goal: Partitioning' is not a valid solution of 'Goal: No_Undesired_Comms'. Instead, what is required to be shown in making the contract is that 'Goal: Partitioning' satisfies 'Goal: No_Undesired_Comms' within the inherited context of 'Goal: No_Undesired_Comms'. It must be possible for the context of participating modules to be simultaneously "true", otherwise the composed argument becomes unsound.

5.5 Incorporating Contract Modules into the Safety Argument Architecture

It is possible to consider the safety argument contract module as part of the safety argument architecture as indicated by the module view in Figure 10.

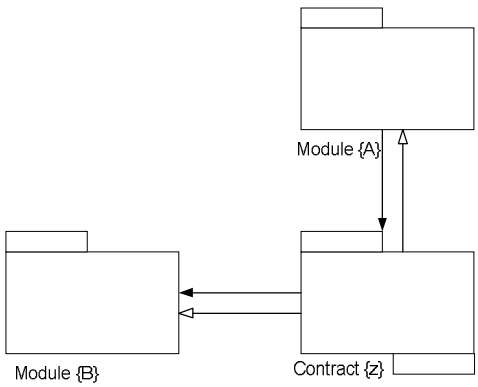

Figure 10 - Including a contract module in the safety argument architecture

Whether the contract modules are represented as an integral part of the architecture or instead, one contract module is produced, with the individual contracts being represented as separate *views* of this module is yet to be resolved, but should not affect the way in which the contracts are developed.

5.6 Notes on Away Goal Decomposition

It should be noted that normally when using GSN it is considered invalid to decompose an away goal. This is because an away goal is merely a reference to a 'real' goal defined elsewhere that may or may not be supported. Therefore, to provide a solution to an away goal is to 'support' a reference to a goal rather than to support the goal itself. In the safety case contract pattern shown in Figure 9, it can clearly be seen that a solution is provided for the away goal referencing the goal requiring support. The purpose of a safety case contract module is specifically to show how a goal *in one module* is supported by a goal *from another module.* In such cases (and only in such cases) we contend that it *is* valid to provide a solution for an away goal. It is important to note that even within a contract module, it is only permissible to decompose an away goal that refers to a goal requiring support. It remains invalid to provide a solution to an away goal that is already supported within its own module, such as goal 'B: Providing Support' in Figure 9. The following property of a safety case module can thus be defined:

- Within safety case contract modules it is valid to decompose away goals which refer to a goal *requiring support* from another module.
- Conversely, the goal requiring support, which is addressed via a contract, must not be decomposed in its host module.

6 Summary

The Ministry of Defence has funded the IAWG to develop a modular safety case and, in doing so, some issues were identified in the definition and use of safety case contracts for modular GSN as defined in Kelly (2001). In discussion with Kelly, IAWG have developed alternate solutions, which IAWG hope will be encompassed within an updated definition of modular GSN in the future. The main points of the alternative solution, as presented in this paper, are summarised below

- A GSN safety case contract module has been proposed as a method of capturing the contract between safety argument modules
- Implementation guidelines for GSN safety contract modules have been developed and are recorded below
- A pattern for GSN safety case contracts is presented in Figure 9
- Assessing and arguing context compatibility is a known and complex issue which requires further work

6.1 Implementation Guidelines

In defining the solution proposed in this paper, a number of process constraints were identified which should be followed when implementing safety case contract modules in GSN. They are summarised below:

- It should not be possible for anyone other than the 'owner' of the safety case module to change the public/private status of a goal.
- It may be necessary to indicate in some way the public goals that are unused for a particular safety case, such that it is clear that the goals are available to other modules, but in this particular case are not required., i.e. provide traceability of public goal usage

- Within safety case contract modules (but *only* within safety case contract modules) it is valid to decompose away goals which refer to a goal requiring support from another module.
- Where a goal requiring support is addressed via a contract, the goal must not be decomposed in its host module.

7 Acknowledgements

The work described here has been undertaken by the Industrial Avionics Working Group, funded by the Ministry of Defence, with support and advice provided by QinetiQ, acting as Independent Safety Advisor, and Tim Kelly of the University of York.

8 References

Kelly, TP (2001). Concepts and Principles of Compositional Safety Cases - (COMSA/2001/1/1) - Research Report commissioned by QinetiQ

Hofmeister, C., Nord, R., Soni, D (1999). Applied Software Architecture, Addison-Wesley