Controlled Expression for Assurance Case Development

Katrina Attwood and Tim Kelly

University of York

York, UK

Abstract *Guidance for developers of assurance arguments has generally focussed on issues concerning content, logical flow and structure. The use of natural language to express an argument can lead to problems with understanding the nature of the claims, with scope and with potentially obscure logical inferences. These problems can occur even if natural language is combined with the use of graphical notations to communicate the structure of an argument. In a supply chain, these problems are compounded by the involvement of numerous suppliers, each with his own "idiolect", which makes it difficult to integrate assurance data for components into the system argument, to evaluate it as evidence or to reuse it across projects. In this paper, we present work to develop controlled language and structured expressions to improve communication within and across domains and to provide some automated validation of assurance arguments.*

1 Introduction

Many industries require an assurance case for safety-critical services, systems and/or software. An assurance case typically comprises reasoned arguments justifying claims relating to the satisfaction of requirements concerning the safety, integrity and/or dependability of a system. These claims are supported by a body of evidence– analysis and test data, design information and process documentation. There is considerable literature providing guidance and methodologies for the development and presentation of assurance cases (e.g. Hawkins and Kelly 2010; Hawkins et al. 2011; Maguire 2006). Much of this guidance focuses on the desirability of presenting **structured assurance cases** (Object Modelling Group 2013), in which the relationships between different claims and between claims and evidence are made explicit. Graphical notations, such as GSN (GSN Community 2011) and CAE (Adelard 2014) have been developed in order to support the development and evaluation of structured assurance cases, although it should be noted that they are not an *a priori* requirement: there is no reason why a structured assurance case should not be presented using natural language alone, and indeed many are. However, guidance on the development of structured assurance cases to date has tended to focus on "macro-level" details of the structure and typical subject-matter of the arguments with relatively little attention paid to the "micro-level" issue of the language used to convey the argument. Issues such as the logical flow and overall readability of the argument – which can be readily summarized using graphical techniques – are foregrounded, while there is only limited guidance on how individual assertions and supporting statements should be phrased to ensure that the argument is correctly conveyed. In the GSN Community Standard (GSN Community 2011), for example, less than 10% of the total document is devoted to language issues.

Imprecise phrasing of claims and assertions can lead to a number of problems with the comprehension of assurance cases:

- **Semantic problems:** An assurance case typically has several authors, each of whom uses language in a distinctive, idiosyncratic way. This means that terms might be used with subtly different meanings at different points in an argument. Similarly, a reader or assessor might not share the writer's understanding of the terms used. This can lead to uncertainties in interpretation, particularly as to what the precise subject of a claim or assertion is or the scope within which the claim/assertion is valid. The author's intended meaning may differ from that conveyed to the audience. A related problem is that of inherent ambiguity, where a given term or phrase has more than one commonly accepted interpretation.
- **Syntactic problems:** Where an argument is carried by claims in 'freeform' text (such as is generally the case even where a graphical notation is used to structure the argument), it may be difficult for a reader to 'unravel' the structure of a sentence so as to establish the scope of the terms used, i.e. how they influence other terms beyond the single phrasal structure in which they occur (Lapore 2009). The relationships between elements under discussion might not be clear, making it difficult for a reader to identify the claims being made and the flow of the argument.

Although these problems may arise with any assurance case, the problems of multiple authorship and readership are compounded in current development practices, where safety-critical systems are increasingly developed by integrating multiple standalone components from a diffuse, multi-organisational, multi-national supply chain. Compositional approaches to certification have been developed for such environments. These require the collection of assurance data relating to discrete components ("Safety Elements Developed Out-of-Context" (ISO/FDIS 2011)). This data then needs to be matched and composed to form an integrated argument for the entire system. Aside from the difficulties inherent in aligning the intent and objectives of assurance requirements in different domains, there is a lower-level need for consistent use of domain- and system-specific terminology across the entire supply chain. This can help in communicating a shared understanding of the nature and limitations of the claims and evidence presented in the argument, and of assumptions made about the operational context in which component behaviour is guaranteed.

One means to address the problems associated with imprecise phrasing is to *constrain* the language used to write assurance cases. For example, a project dictionary can be used to address semantic issues. The dictionary declares the accepted meaning of terms that are inherently ambiguous or often misunderstood, or which may be commonly used with subtly different meanings in different usage contexts. There are several advantages to this approach, particularly in ensuring that the meaning intended by an author is that received by the reader, and also in making it easier to "match" terminology used in assurance data for discrete components, by making similarities and differences in meaning clear. However, there are considerable overheads in the initial development – and agreement – of the domain dictionary, and also in ensuring that it is adhered to. Constraining the syntax of assertions made in the assurance case – for example, by providing structured expressions – can help address uncertainties of linguistic scope and uncertainty. These are instantiatable "patterns" for sentence structures, which define the range of linguistic variables which can occur in particular relations. Although certain generic types of expression can be identified in the assurance case domain (see Attwood et al. forthcoming), for the most traction to be gained from this method specialised patterns for each domain are required. For this, we require a thorough understanding of the requirements for assurance in a particular domain, to ensure sufficient focus and relevance in the argument claims generated using the controlled expressions.

In this paper, we outline an approach to controlled language and expression for assurance cases which addresses the semantic and syntactic issues associated with natural language identified above. This approach has been developed on the EC-funded OPENCOSS project (OPENCOSS Consortium 2011), where it augments a model-based conceptualization of safety assurance which is used to address issues of cross-domain and cross-project reuse of safety assurance assets, including arguments. In principle, however, the language approach can be standalone, as it is presented here. Section 2 outlines the usage scenarios for controlled language identified in the OPENCOSS project. Section 3 presents conceptual metamodels to address each of these usage scenarios and encapsulate the proposed approach. Section 4 presents a brief worked example of the use of controlled vocabulary and structured expressions to develop an assurance argument in the automotive domain.

2 Usage Scenarios

In this section, we describe three basic usage scenarios for controlled language in the development of assurance cases. The general context for this work is a compositional certification context, where a system-level assurance case is being assembled by an integrator organisation from assurance data relating to freestanding, pre-existing components. In some cases, the components may have been developed within another safety-critical domain and are being reused in the integrated system (for example, a software module originally developed for an aerospace application is being reused in a rail system). Note that this paper does not address the complex technical implications of developing a compositional safety case, but is concerned simply with *linguistic aspects* of the project: principally, our concerns are to ensure that there is a clear, shared understanding of the meaning and scope of domain- and system-specific terminology throughout the assurance case, so that the technical aspects of integration of assurance case data can be assessed from a common basis. In particular, we are concerned that the differences in terminology across domains are clearly documented and the implications of reuse understood.

2.1 Development of Structured Vocabularies for Assurance

The development of vocabularies to provide constrained definitions of the terminology used in assurance cases is the prerequisite for the other usage scenarios outlined in this section. Section 3.1 below outlines a conceptual metamodel for the development of such vocabularies for each of the domains and projects relevant to the assurance case (e.g. for rail and avionics). The vocabularies are structured as concept hierarchies, along the lines of an English language Thesaurus (for example (Lloyd 1982)), with terms being grouped according to general concept types. A concept type is a general category, by which related concepts at lower levels of detail can be grouped. For example, in a vocabulary developed for the automotive domain using ISO 26262 (ISO/FDIS 2011) the terms “driver”, “passenger” and “cyclist” are all grouped under the concept type “traffic participant”, which is itself grouped under the general concept “agent”. Definitions are provided in a semi-formal structured English, derived from Semantics of Business Vocabulary and Business Rules (SBVR) (Object Modelling Group 2008). SBVR is an OMG standard for the preparation of controlled or semi-controlled vocabularies for a given domain and for modelling the relationships between domain concepts in intuitive natural language. Where terms defined elsewhere in the vocabulary are used in the definitions, cross-references are provided. In order to facilitate comparison between terminology across domains, concept types are defined fairly generically across the domains, resulting in a commonality of general structure, even though the low-level terms grouped under each generic concept type might differ substantially between domains. Linguistic relations are used to indicate relationships between terms, such as “is-a” and “part-of”. We develop vocabularies at two levels of abstraction: domain-specific vocabularies, which are derived from the terminology used in the relevant safety standards for the domain, and project-specific vocabularies, which specialise the domain vocabulary for a given project or organisation.

Having clear definitions of terminology in which concepts are related both vertically by type and sub-type relations and horizontally by being defined in terms of one another can help in ensuring consistency of reference across assurance cases. In particular, the terminology can be used to characterise the “interfaces” between concerns in an assurance argument (which may be represented explicitly in a modular assurance case), to ensure that the terms of reference are consistently understood across a supply chain.

2.2 Structured Expressions in Assurance Arguments

One important means of maintaining consistency in the natural language used to convey reasoning in an assurance argument is to regularize the structure of the statements used to express claims. Having a common syntactic structure makes it easier for a reader to parse claims, and avoids issues such as confusion over the scope of a given term in a sentence. It is quite common in “engineering prose” for there to be uncertainties over the interpretation of the scope of qualifiers, as in the phrase “failure mode and effect analysis”, where “analysis” serves to qualify both “failure mode” and “effect” and where “failure” qualifies both “mode” and “effect”: a non-specialist reader would be likely to look only for the most limited range, and associate “analysis” only with “effect” and “failure” only with “mode”.

Structured expressions can be used to characterise the types of concepts which are discussed at a particular point in an argument and the relevant features which can be asserted about them. Typically, a structured expression comprises a “fixed” verb phrase, which carries the burden of the claim, while noun phrases, providing the subject and object over which the verb phrase ranges, are parameterisable. For example, a very simple structured expression in an assurance case claim might take the form “{systematic fault} is adequately mitigated by {fault mitigation technique}”, where both “systematic fault” and “fault mitigation technique” are broad parameters. Simple expressions can be combined to form larger syntactic units, for example: “{systematic fault} is adequately mitigated by {fault mitigation technique} which addresses {hazard}”.

In our approach, a series of generic structured expressions are defined, and used to refine the logical structures summarized in argument fragment templates captured in GSN patterns such as those in (Hawkins and Kelly 2013), by specifying the types of concepts which are in focus at particular points in the argument. The generic concept types which are used to structure the vocabularies (see section 2.1) are used to reference the variable parameters, e.g. ‘fault”, “fault mitigation technique", "hazard" are all high-level concept types common across the domain vocabularies. These expressions can then be instantiated to form claims in specific arguments, by supplying variables from the terms listed under the relevant concept types as appropriate. Since we have both domain- and project-specific vocabularies, we can instantiate claims at two levels of abstraction, meaning that we can use the concept types to specify detailed patterns at the domain level, for instantiation by a project. Automated support for this stage can be provided, with the concept types in the vocabulary models being used to present a series of potential instantiations of a given parameter from which an argument writer can choose.

2.3 Cross-Domain Vocabulary Mapping

As described above, the general context for our work involves the creation of a compositional assurance case from pre-existing components, some of which are being reused across domains. This cross-domain scenario intensifies the semantic problems concerning shared understanding of terminology observed in Section 1. There is no shared conceptualisation of assurance across safety-critical domains and the objectives and requirements of certification approaches differ fundamentally. Even where terms in different domains may appear similar, the concepts they represent may not be. As a motivating example, we consider the difficulties of reusing software developed according to IEC 61508 (IEC 2009) in an avionics context, where certification to DO-178B (RTCA 1992) is required. An assurance argument in the original context might assert that "software module Y is developed to safety integrity level SIL 4". In the avionics context, the manufacturer might wish to make a similar claim: "software component Y is developed to design assurance level DAL A". Since both the safety integrity level and the design assurance level are instantiations of the generic concept type "criticality level", it might be assumed that a direct replacement of one term for the other is allowable. Closer examination of the definitions of the terms, however, will reveal an important distinction between the two level descriptors. In IEC 61508, a SIL is directly associated with a (software) safety function which is modelled at the system level. In DO-178B, however, the DAL is associated with a software system or component and does not address the "function" concept at all. It is not possible to convert a SIL directly into a DAL without considering the extra process-related requirements that arise because of DO-178B's focus on the design of the system, rather than merely its functionality. What is required here is not a definition of individual concepts in isolation, but an appreciation of the interrelationships between the concepts, since these provide constraints on the reuse of the claim and associated assurance data.

We aim to provide a mechanism for informed "translation" between terminology used in assurance in different safety-critical domains, in such a way that the different understandings and scope of assurance concepts within the different domains are clearly documented and understood. Initially, we do this by providing guidance to a user in asserting mappings between concepts presented in the domain-specific vocabularies for the source and target domains. The metamodel underpinning this approach is presented in Section 3.2 below.

Three general classes of mapping are identified – exact map, partial map and no map. In the majority of cases, exact mappings of terms will not be possible across domains – instead, the mappings will need to be partial. The model allows for a distinction between "broad" and "narrow" partial mappings. Concepts in the vocabularies are organised hierarchically, using generic concept types. Initially, candidate mappings are presented to the user by simply displaying all of the elements in the vocabulary for the target domain which are categorized using the same concept type(s) as the source term. Although the source and target domain vocabularies are structured using a standard set of concept types, there may be a considerable number of candidate maps. The field is then narrowed by displaying the definitions for each of the candidate maps. The user can use these definitions to make an informed choice as to the most suitable replacement term, based on the definition, any relationships implied by the use of reserved terms in the definition and the degree of coverage of the source term in the chosen target. This information is recorded manually in a "map justification", so that the rationale for the reuse and any limitations on it are preserved.

3 Conceptual Metamodel

In this section, we present the conceptual metamodel which underpins our approach to structured language in assurance cases. The conceptual model has been heavily influenced by several existing models for concept definition and thesaurus development:

- **OMG Structured Assurance Case Metamodel (SACM)** (Object Modelling Group 2013) - an OMG standard which defines the treatment of argumentation and evidence in structured assurance cases and provides a common interchange format. The first version of SACM was published in 2013. Proposed revisions will be discussed in December 2014 with a view to a second version shortly thereafter.
- **Simple Knowledge Organization System (SKOS**) (W3 Consortium 2009) – a W3C recommendation for the representation of Thesauri and other classification schemes. SKOS was developed and refined in three EC-funded projects in the late 1990s and 2000s, and has some industry support. Its primary significance here is in the concepts it provides for the development of concept hierarchies linked by non-hierarchical relationships and in its introduction of the use of "labels" to support the recording of full synonyms.
- **ISO 25964:2011 Thesauri and Interoperability with Other Vocabularies** (ISO 2011) - This ISO standard provides guidance on mapping

between one vocabulary and another, as well as providing a series of relationships between concepts and terms (equivalence relationships, hierarchical relationships and associative relationships).

- **OMG Ontology Definition Metamodel** (Object Modelling Group 2009) - This document provides metamodels for three leading ontology representation paradigms – Common Logic, RDF and OWL, and explains their relationship with and divergences from UML.
- **OMG Semantics of Business Vocabulary and Business Rules (SBVR)** (Object Modelling Group 2008) - an OMG standard for the preparation of controlled or semi-controlled vocabularies for a given domain and form modelling the relationships between them in intuitive natural language. SBVR provides a mechanism for the presentation of "reserved terms" in semi-formal definitions of concepts and also for the super- and sub-typing of concepts in concept hierarchies. SBVR "fact types" provide a basis for some of the controlled expressions used in assurance arguments.

For ease of presentation, the metamodel is presented in three views. Section 3.1 describes basic vocabulary metamodel. Section 3.2 extends this model to include concepts required to support structured expressions in assurance arguments. Section 3.3 provides the mapping metamodel required to support cross-domain vocabulary "translation".

3.1 Vocabulary Metamodel

Figure 1 depicts the conceptual vocabulary metamodel developed in OPENCOSS. The notation used here and in Figures 2 and 3 below is Ecore, the semantics of which accord with the UML[1]. This model defines the concepts and relationships required for the development of structured vocabularies for safety-critical domains and projects, as outlined in Section 2.1. Textual definitions of the core elements of the model are presented below.

[1] Ecore is the foundational metamodel of the OMG's Meta-Object Facility standard for model-driven engineering (http://www.omg.org/mof/). The UML metamodel is defined in Ecore.

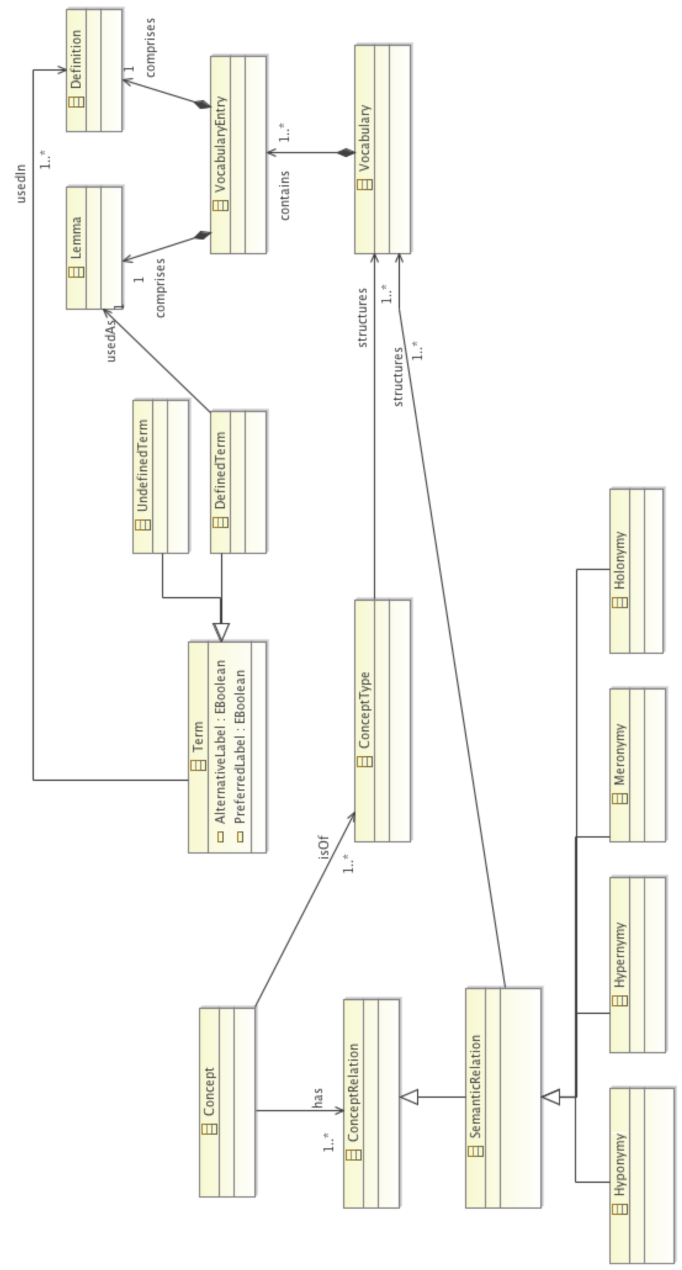

Fig. 1: Conceptual Vocabulary Metamodel

Concept: Any unit of meaning which can be described or defined by a unique combination of characteristics (Object Modelling Group 2008; W3 Consortium 2009). A Concept is represented/reified by the *Term* class and defined by the *Definition* class.

ConceptType: A container class which classifies things on the basis of their similarities. Each *Concept* belongs to one or more *ConceptType*. This is akin to the "ConceptType" relation in SBVR (Object Modelling Group 2008). The *ConceptTypes* provide the basic structure of the *Vocabulary* in that *Terms* are arranged according to the *ConceptTypes* of the *Concepts* they represent.

ConceptRelation: A container for the types of relation between *Concepts*, which are used to model the comparison between two *Concepts* in two different *Vocabularies* or to structure a *Vocabulary* in terms of the relationships between *Concepts* grouped within a *ConceptType*. Two types of *ConceptRelation* are identified: *MappingRelation* (discussed in Section 3.3) and *SemanticRelation*.

SemanticRelation: A container for the finer relationships between the meanings of *Concepts* within a *ConceptType,* used to structure *Terms* of the same *ConceptType* in a *Vocabulary*. *SemanticRelations* are kept separate from *MappingRelations* in this conceptual model, in order to highlight their linguistic basis, as opposed to the logical ones described by *MappingRelations*. *MappingRelations* are described in section 3.3 below.

Hyponymy: A *SemanticRelation* by which the meaning of one *Concept* is more specific than that of another *Concept*. In other words, a hyponym is used to designate a member of a general class. For example, 'systematic fault' and 'intermittent fault' are both hyponyms of 'fault'. This might be referred to as a "type-of" relationship.

Hypernymy: A *SemanticRelation* by which the meaning of one *Concept* is more general than another *Concept*. In other words, a hypernym designates the general category, of which its hyponyms are members or subdivisions. For example, 'fault' is a hypernym of 'systematic fault' and 'intermittent fault'. This might be referred to as a "supertype-of" relationship.

Meronymy: A *SemanticRelation* by which one *Concept* is a constituent part of a general whole captured in another *Concept*. For example, "wheel" is a meronym of "automobile". This might be referred to as a "part-of" relationship.

Holonymy: A *SemanticRelation* by which one *Concept* is an aggregation of other *Concepts*. For example, "automobile" is a holonym of "wheel", "chassis" etc. This might be referred to as a "contains" relationship.

Term: The word or phrase which represents (or reifies) a *Concept*, typically a noun or noun-phrase (ISO 2011). *Terms* are thus the basic domain vocabulary, and are stored in the *Vocabulary*. A *Term* may be thought of as providing a label for a *Concept*, an unambiguous means by which the *Concept* can be referenced. Each *Term* has two Boolean attributes, *PreferredLabel* and *AlternativeLabel*. If the *PreferredLabel* attribute is set true, then the *Term* serves as the primary means by which a *Concept* is referred to and the principal key to represent that *Concept* in the *Vocabulary.* If the *AlternativeLabel* is set true, the term serves as an alternate means to reference the *Concept*, there being a *PreferredLabel* elsewhere in the *Vocabulary*. This mechanism allows for the unambiguous recording of exact synonym relationships between terms.

DefinedTerm and UndefinedTerm: We identify two subtypes of *Term*. *DefinedTerms* are those which are included in the *Vocabulary* and which should be used with a single "reserved" meaning in project artefacts. Each *DefinedTerm* is represented in the *Vocabulary* by a *VocabularyEntry*. *UndefinedTerms* are those which have no "reserved" meaning in the project, and which are not therefore defined in the *Vocabulary*. *Terms* of all kinds can be used in *Definitions*.

Vocabulary: An aggregation of the *VocabularyEntries* which identify and define *DefinedTerms* for a particular domain. The broad structure of the *Vocabulary* is provided by the *ConceptTypes*, and relationships between the *Concepts* represented by *DefinedTerms* are captured by *SemanticRelations*.

VocabularyEntry: Each *DefinedTerm* has a *VocabularyEntry* which encapsulates the information stored concerning the *DefinedTerm* in the *Vocabulary*. Each *VocabularyEntry* comprises exactly one *Lemma* and exactly one *Definition*.

Lemma: A string representing the noun or noun-phrased used to designate a *DefinedTerm*. The *Lemma* serves as the key to identify the *DefinedTerm* in the *Vocabulary*.

Definition: A string which contains an unambiguous description of the *Concept* represented by a *DefinedTerm*. The nouns and noun-phrases used in *Definitions* may be either *DefinedTerms* or *UndefinedTerms*.

3.2 Metamodel for Structured Assertions

Figure 2 depicts the Ecore metamodel for structured assertions. This model derives from proposed modifications to the SACM (Object Modelling Group 2013), which have been made by the SACM Revision Task Force, and will be discussed at length at the OMG in December 2014[1]. This metamodel defines the use of controlled language in elements of assurance arguments. Two aspects of language constraint are addressed. Individual words and phrases are used with fixed definitions, as described in Section 3.1, to refer to concrete declared objects (the subjects of discussion in the assurance case). As described above, these terms and phrases are *DefinedTerms*, contained in a *Vocabulary*. Where structured assertions are used, these has a *SentenceStructure*, in which each term has a fixed *Role*, a clear function in the syntax and semantics of the sentence. *RoleBindings* link instances of these *Roles* to actual instances, which are used in *Assertions* in the argument. Textual definitions for the core elements of the model are given below.

[1] Note that this is an abbreviated version of the proposed model, in which only elements of interest to the present discussion have been included. Other subclasses of *ExpressionElement* and *ArgumentElement* occur in the SACM, but have been omitted here for clarity.

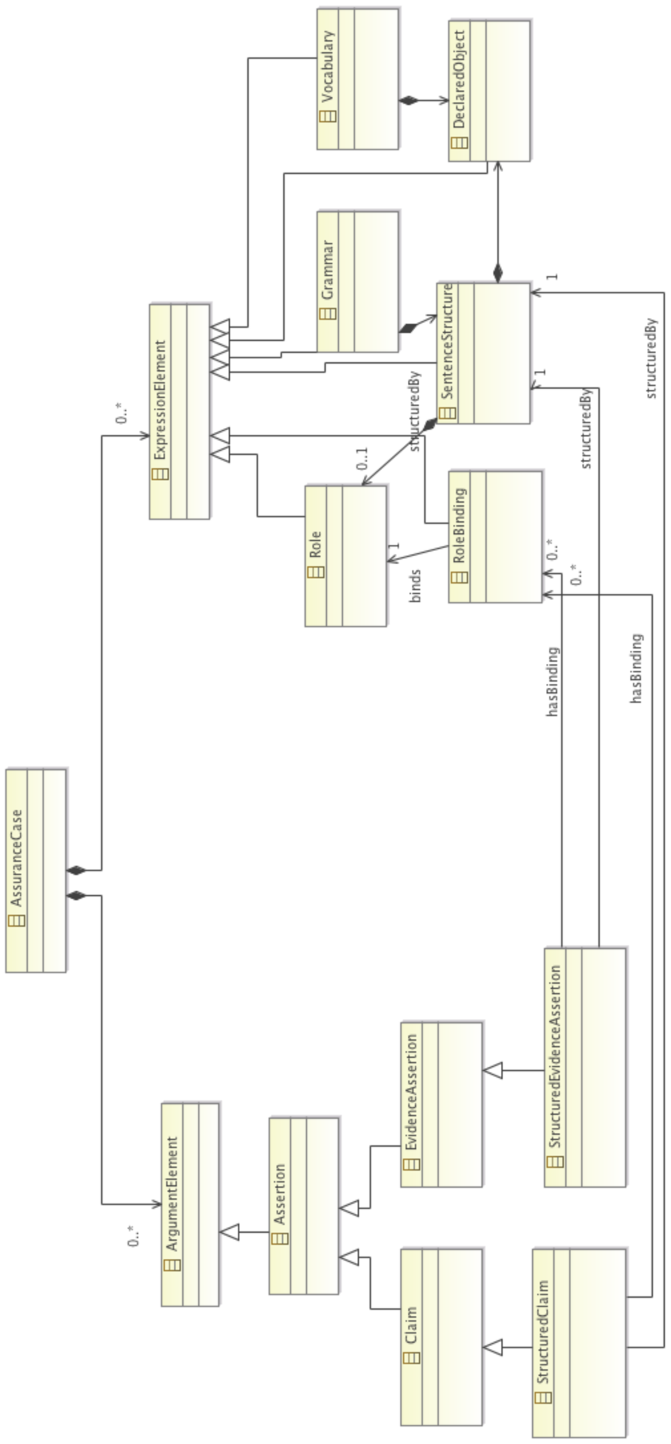

Fig. 2: Conceptual Structured Assertions Metamodel

AssuranceCase: A container class for elements comprising a structured assurance case: claims, argumentation and evidence which seek to establish that a given system satisfies relevant requirements.

ExpressionElement: A container class for those aspects of an *AssuranceCase* which are associated with the way in which the argument is expressed in natural language.

Vocabulary: An aggregation of *VocabularyEntries* (see section 3.1 above) relating to *DefinedTerms*.

DeclaredObject: A reference to an instance of a *Concept* to which a *DefinedTerm* refers. See section 3.1 above.

Grammar: An aggregation of *SentenceStructures*, representing the total of permissible structured expression types for propositions to be used in a given *AssuranceArgument.*

SentenceStructure: A pattern for a structured expression, which relates *DeclaredObjects* to one another in a proposition, using controlled syntactic structures in which the function of each *DeclaredObject* is clarified by the use of a *Role*.

Role: A description of the function played by a given *DeclaredObject* in the relationship captured in a *SentenceStructure*. Grammatically, a *DeclaredObject* will be either the subject or direct object of a verb. However, the *SentenceStructures* defined for *AssuranceCases* are more sophisticated, in that the verbs “dictate” the *DeclaredObjects* which can participate in the relationship. Hence, a *Role* is a clear description of the function of a *DeclaredObject* in the system or its assurance.

RoleBinding: The connection of a given *Role* to an actual variable, expression or concept which can be used to instantiate it.

ArgumentElement: A container class for those aspects of an *AssuranceCase* which are associated with the logical structure of the argumentation or the presentation of evidence.

Assertion: An abstract class to capture types of propositions which can be used in an argument.

Claim: A natural language statement which is used to record a proposition used as a premise or a conclusion in an argument, to express an opinion or judgment.

StructuredClaim: A *Claim* whose expression is an instantiation of a pattern captured in a *SentenceStructure*.

EvidenceAssertion: A natural language statement which is used to record a proposition concerning some quality or characteristic of an item of evidence or its relationship with some other element discussed in the argument.

StructuredEvidenceAssertion: An *EvidenceAssertion* whose expression is an instantiation of a pattern captured in a *SentenceStructure.*

3.3 Mappings Metamodel

Figure 3 depicts the conceptual mappings metamodel developed in OPENCOSS. This model defines the concepts and relationships required to support the technique to provide guidance for cross-domain vocabulary "translation", as described in Section 2.3. Subjective mappings of different degrees of exactness between concepts are identified, based on the degree of "overlap" between the meanings and usage of concepts in the different domains. In the OPENCOSS tooling, mappings are asserted manually, and a textual "Map Justification" is supplied, to describe the nature of the mapping, capturing the similarities and differences between the concepts. It is particularly important that *NoMap* relationships between concepts which might appear similar are recorded where "translation" is inappropriate. Textual definitions for the core elements of the model are given below.

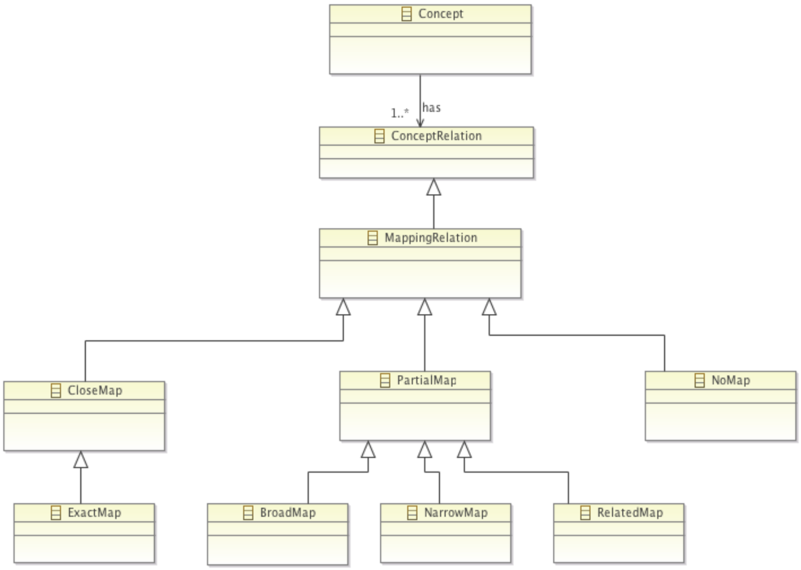

Fig. 3: Conceptual Mappings Metamodel

Concept: See section 3.1 above.

ConceptRelation: See section 3.1 above.

MappingRelation: A container for the relationships between the meanings and contexts of use of two *Concepts* represented by *Terms* in two different *Vocabularies* (across domains or projects).

CloseMap: A *MappingRelation* which indicates that there is a good degree of similarity between the meanings and contexts of use of two *Concepts* which are being compared across two *Vocabularies*.

ExactMap: A *CloseMap* which indicates that the relationship between two *Concepts* is such that the meaning and context of the target *Concept* is identical with that of the source *Concept.*

PartialMap: A *MappingRelation* which indicates that there is some similarity between the meanings and contexts of use of two *Concepts* which are being compared across two *Vocabularies*, but that the similarity is not sufficient to reach the threshold for a *CloseMap.*

BroadMap: A *PartialMap* in which the meanings and contexts of use of the two *Concepts* can be considered similar in terms either of important attributes or the proportion of overall similarity.

NarrowMap: A *PartialMap* in which the meanings and concepts of use of the two *Concepts* can be considered less similar in terms either of important attributes or the proportion of overall similarity than is the case for a *BroadMap*.

RelatedMap: A *PartialMap* which indicates that the meanings and concepts of use of the two *Concepts* can be considered to be related but there is no conceptual overlap between them.

NoMap: A *MappingRelation* which asserts that there is no similarity or overlap between the meanings and contexts of use of two *Concepts* which are being compared across two *Vocabularies,* with respect to features considered relevant for asserting a mapping.

4 Example

In this section, we present a simple example to illustrate the ways in which structured expressions using controlled vocabulary can be exploited to instantiate claims in an assurance argument. The example is based on a simplified, fictitious automotive anti-lock braking system (ABS), which is developed to ISO 26262 (ISO/FDIS 2011). Correct operation of the ABS allows the wheels to maintain contact with the road surface during hard braking, preventing the wheels from locking and avoiding an uncontrolled skid. The system comprises a software controller, four wheel sensors (one for each wheel) and two hydraulic valves (one for each axle). The system has two basic operational scenarios. The software monitors the speed at which the wheels rotate, measured via the wheel sensors. If it detects that one wheel is rotating at a slower speed than the others, the controller actuates the hydraulic valves to reduce hydraulic pressure to the brake, thus reducing braking force on that wheel and allowing it to turn faster. Alternatively, if the software detects that one wheel is turning significantly faster than the others, the valves are operated to increase hydraulic pressure to that wheel, thus increasing braking force to that wheel and slowing down its rotation. The software controller contains a critical function to calculate the hydraulic pressure demand value from the wheel speed sensor inputs. Failure of this function results in the incorrect braking force being applied to the wheel, which could result in a skid.

The assurance argument for the ABS software controller needs to address the issue of potential faults in the hydraulic pressure demand calculation function. In this example, that issue will be addressed as part of a top-down argument concerning the mitigation of the "uncontrolled skid" hazard by the software. An argument of this type can be structured using the high-level software safety argument pattern in (Hawkins and Kelly 2013), which is presented in Figure 4, using the GSN (GSN Community 2011; Kelly 1998). High-level patterns for the expression of the structured assertions are captured in this pattern. In terms of the metamodels presented in sections 3.1 and 3.2 above, *Roles* are captured as parameterisable noun-phrases contained in {}, while the *SentenceStructure* is captured as the entirety of the phraseology contained within a given GSN claim. The totality of *SentenceStructures* in the diagram represents the *Grammar* for this particular pattern.

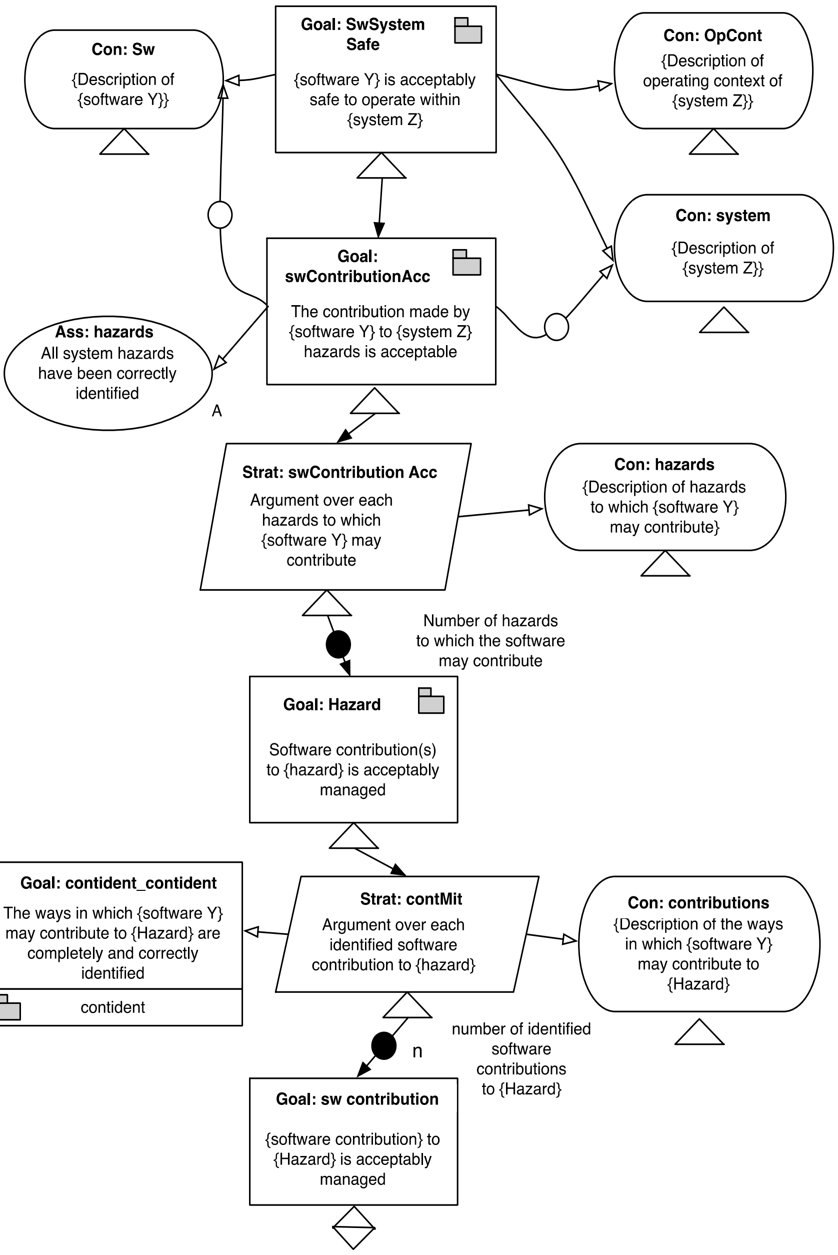
Goal: SwSystem Safe
{software Y} is acceptably safe to operate within {system Z}
Con: Sw
{Description of {software Y}}
Con: OpCont
{Description of operating context of {system Z}}
Con: system
{Description of {system Z}}
Goal: swContributionAcc
The contribution made by {software Y} to {system Z} hazards is acceptable
Ass: hazards
All system hazards have been correctly identified
A
Strat: swContribution Acc
Argument over each hazards to which {software Y} may contribute
Con: hazards
{Description of hazards to which {software Y} may contribute}
Number of hazards to which the software may contribute
Goal: Hazard
Software contribution(s) to {hazard} is acceptably managed
Goal: contident_contident
The ways in which {software Y} may contribute to {Hazard} are completely and correctly identified
contident
Strat: contMit
Argument over each identified software contribution to {hazard}
Con: contributions
{Description of the ways in which {software Y} may contribute to {Hazard}
n
number of identified software contributions to {Hazard}
Goal: sw contribution
{software contribution} to {Hazard} is acceptably managed

Fig. 4 High-Level software safety argument pattern (from (Hawkins and Kelly 2013))

Our discussion draws on the lower part of the pattern in Figure 4, the claim in Goal:Hazard that the software's contribution to a particular Hazard is adequately managed and the subsequent argument addressing each potential way in which the software could contribute to the hazard.

The example requires two distinct *Vocabularies* containing *VocabularyEntries* by which *DefinedTerms* are grouped initially as abstract ConceptTypes and then as concrete instances of these types relating to particular *DeclaredObjects* in the ABS project documentation. Firstly, the ABS System is represented in a vocabulary, terms in which are drawn from the organisational vocabulary for the system as a whole and which define concepts in the deployment context of the ABS software. Concepts relating to the structure and functionality of the ABS software are also represented in a dedicated, project-level, vocabulary.

Figure 5 shows a partial instantiation of the template pattern presented in Figure 3, as an assurance argument for the ABS Software. The underlined terms here ("ABS software", "uncontrolled skid hazard", "safety requirement 123", "fault tree analysis") are *DefinedTerms* in the vocabulary for the ABS System (populated from project documents at the system level, such as system descriptions, requirements documents, system safety analysis), and relate to *DeclaredObjects* which fulfil the *Roles* indicated in the *SentenceStructures* in Figure 4.

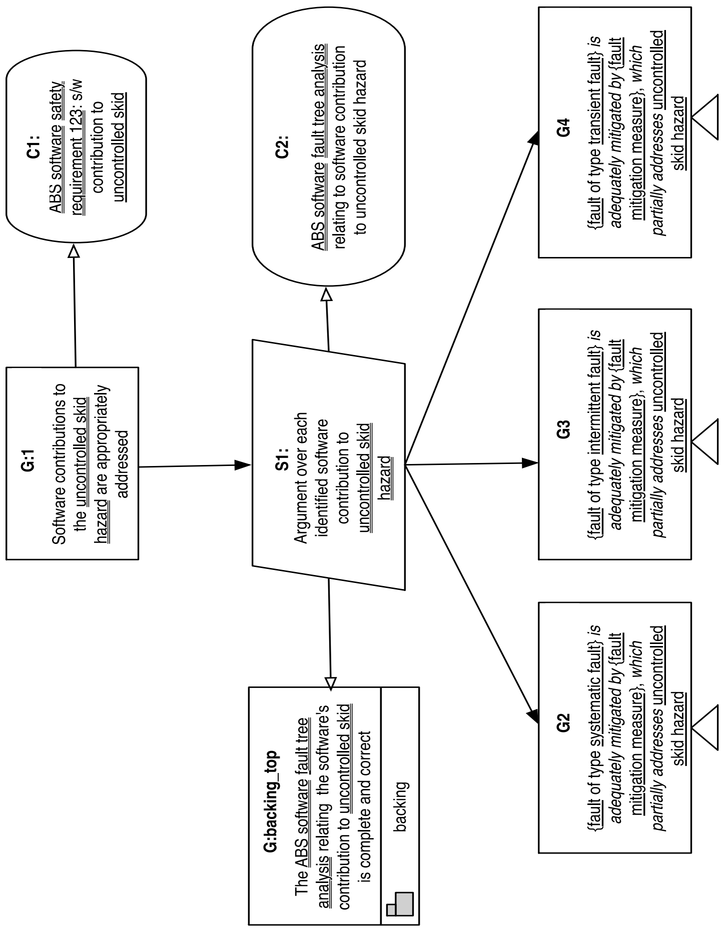
C1:
ABS software safety requirement 123: s/w contribution to uncontrolled skid
G:1
Software contributions to the uncontrolled skid hazard are appropriately addressed
C2:
ABS software fault tree analysis relating to software contribution to uncontrolled skid hazard
S1:
Argument over each identified software contribution to uncontrolled skid hazard
G:backing_top
The ABS software fault tree analysis relating the software's contribution to uncontrolled skid is complete and correct
backing
G4
{fault of type transient fault} is adequately mitigated by {fault mitigation measure}, which partially addresses uncontrolled skid hazard
G3
{fault of type intermittent fault} is adequately mitigated by {fault mitigation measure}, which partially addresses uncontrolled skid hazard
G2
{fault of type systematic fault} is adequately mitigated by {fault mitigation measure}, which partially addresses uncontrolled skid hazard

Fig. 5 Restatement of lower portion of software safety argument pattern

It will be clear that the first part of the claims in Goals G2, G3 and G4 are represented here as a less abstract *SentenceStructure* than in Figure 4, but that the *RoleBinding* to a definite *DeclaredObject* is not yet complete. Instead, they assert the relationship which is modelled between two *ConceptTypes* in the *Vocabulary*: “fault” and “safety measure”. This relationship is captured in the *SentenceStructure* safety measure *mitigates* fault. The second part of the claim is generated directly from a *RoleBinding* linking a specific *DeclaredObject* to the ConceptType {Hazard} presented in Figure 4. Further traversals of the *ConceptTypes* presented in the *Vocabulary* are required to instantiate the *RoleBinding* for the first part of the claim, as shown in the partial instantiation of Goal G2 as two parallel goals addressing systematic faults:

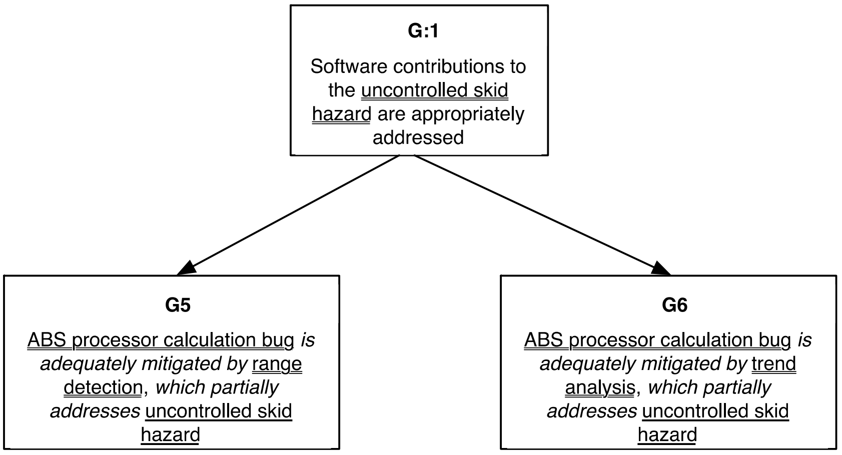
G:1
Software contributions to the uncontrolled skid hazard are appropriately addressed
G5
ABS processor calculation bug *is adequately mitigated by* range detection, *which partially addresses* uncontrolled skid hazard
G6
ABS processor calculation bug *is adequately mitigated by* trend analysis, *which partially addresses* uncontrolled skid hazard

Fig. 6 Partial instantiation of Goal G2

Here, Goal G2 from Figure 4 has been instantiated twice, populated using instances of the “systematic fault” and “fault mitigation measures” *ConceptTypes* (a synonym for “safety measure”) from the *Vocabulary* for the actual ABS system as *DeclaredObjects* fulfilling the *Roles* indicated in Figures 4 and 5. Note that the intention here is to show the population of the generic claim type using *DefinedTerms* from the vocabulary, rather than to present a complete argument.

5 Conclusion and Further Work

This paper has demonstrated the potential use of controlled vocabulary and structured expressions to add rigour to the language used to convey assurance arguments for safety-critical systems. We have presented metamodels for the creation of structured vocabularies, for controlled expressions and for cross-domain comparison of terminology, and have demonstrated how controlled expressions can be instantiated to inform the reuse of assurance assets. Work to develop this method and to provide tooling is currently at a relatively early stage. Further work is required to define structured vocabularies for the OPENCOSS target domains (rail,, automotive and aerospace) and to agree a common hierarchy of *ConceptTypes* which can be used to structure and compare them. In principle, assuming that there is a degree of consistency in the terminology used for *ConceptTypes* and *Definitions* across the safety-critical domains, automated support could be provided for the cross-domain "translation" of terminology. This could be achieved by providing information about the relative positions of source and candidate target terms in the concept hierarchy provided by the *ConceptTypes* (i.e. the conceptual distance between a *Term* and a generic concept and the nature of the mappings between interim terms) and the relative similarity of their *Definitions*. In practice, it is unlikely that such a technique could ever replace manually-asserted mappings between vocabulary terms: at most, they could provide a means for narrowing and prioritizing the list of candidate mappings. The technique could also be deployed to provide informed "translation" of generic assurance argument patterns into domain-specific versions, by the tailoring of vocabulary to suit the application context.

Acknowledgments The work presented here was carried out as part of the OPENCOSS Project, No:289011, which is funded by the European Commission under the FP7-ICR Framework. For further details, see the Project website: http://www.opencoss-project.eu.

References

Adelard (2014) Claims Argument Evidence Notation. http://www.adelard.com/asce/choosing-asce/cae.html. Accessed 30 September 2014

Attwood K, Conmy P, Kelly T (forthcoming) The use of controlled vocabularies and structured expressions in the assurance of CPS. Ada Users' Journal 2014

GSN Community (2011) Goal Structuring Notation Community Standard. Issue 1. http://www.goalstructuringnotation.info. Accessed 1 October 2014

Hawkins R, Kelly T (2010) A systematic approach to developing software safety cases. J System Safety 46: 25-33

Hawkins R, Kelly T (2013) A software safety argument pattern catalogue. University of York. ftp://ftp.cs.york.ac.uk/reports/2013/YCS/482/YCS-2013-482.pdf, Accessed 25 September 2014

Hawkins R, Kelly T, Knight J, Graydon P (2011), A new approach for creating clear safety arguments. In Dale C, Anderson T (eds) Advances in systems safety. Springer

IEC (2009) IEC 61508: International standard – functional safety of electrical/electronic/programmable electronic safety-related systems

ISO (2011) ISO 25964 International standard for thesauri and interoperability with other vocabularies.

ISO/FDIS (2011) ISO/FDIS 26262 International standard – road vehicles, functional safety

Kelly T (1998) Arguing safety – a systematic approach to managing safety cases. DPhil Thesis. University of York.

Lapore E (2009) Meaning and argument. John Wiley and Sons

Lloyd S (ed) (1982), Roget's Thesaurus of English words and phrases. Penguin

Maguire R (2006), Safety cases and safety reports: meaning, motivation and management. Ashgate.

Object Modelling Group (2008) Semantics of Business Vocabulary and Business Rules (SBVR). Version 1. http://www.omg.org/spec/SBVR/1.0. Accessed 25 September 2014

Object Modelling Group (2009) Ontology Definition Metamodel. Version 1. http://www.omg.org/spec/ODM/1.0. Accessed 24 June 2014

Object Modelling Group (2013) Structured Assurance Case Metamodel (SACM). Version 1. http://www.omg.org/specs/SACM. Accessed 3 October 2014

OPENCOSS Consortium (2011) http://www.opencoss-project.eu. Accessed 3 October 2014

RTCA (1992) RTCA/DO-178B: Software considerations in airborne systems and equipment certification

W3 Consortium (2009) Simple Knowledge Organization System (SKOS) Reference. http://www.w3.org/TR/2009/REC-skos-reference-20090818. Accessed 24 June 2014