Nuanced Term-Matching to Assist in Compositional Safety Assurance

Katrina Attwood and Philippa Conmy
Department of Computer Science
University of York
Heslington, U.K.
{katrina.attwood, philippa.conmy}@york.ac.uk

***Abstract*—Increased complexity in the design, technology and supply chains for software-intensive safety-critical systems has resulted in a growing demand for a compositional approach to safety assurance. Assurance data relating to independently-derived components must be melded together into a compelling case for overall system safety. One of the barriers to composition is the lack of consistency in the terminology used to describe and share assurance data. Linguistic mismatches highlight various problems for the composition of peer modules and their integration into an overall case. In this paper, we propose the application of a linguistic model of understanding to identify mismatches and to provide guidance on composition and integration. The approach is illustrated using a simple example.**

***Index Terms*—Compositional safety assurance, component, natural language, Goal Structuring Notation**

I. Introduction

Software-intensive safety-critical systems are becoming increasingly complex, made from multiple components from many different suppliers. Partly this is due to general trends in technology, making it more economic to buy off-the-shelf consumer components (such as processors or software libraries). These may not be ideally suited to dependable systems, but bespoke alternatives are too costly to consider. Systems are also becoming more complex, with increased functionality, and upgrades (or maintenance) implemented at a faster pace. This may be in domains as diverse as the military (using systems of systems of devices and components from trusted and untrusted sources) and automotive (with owners needing software updates to deal with new functionality, fix bugs and so on).

The increased complexity in design means that it is increasingly difficult to assess whether a composed system is fit for purpose, i.e. acceptably safe and functional. Typically, a safety-critical system undergoes some sort of approval or certification process to assess and demonstrate its safety properties. Numerous sets of guidance and standards exist to assist in this process, but most assume that a bespoke system is being developed using a rigid top-down design process, rather than including off-the-shelf (OTS) components composed together. This means that the analysis techniques they contain may be difficult to apply independently, or to meld together in a way that is compelling or useful to demonstrate safety. Newer standards and methodologies contain some approaches for the composition of certification data but there are still many unanswered questions as to how to do this practically, balancing the need to demonstrate safety against cost, improved speed for design or maintenance and technical challenges when composing analysis results.

One of the barriers to the composition of safety argument modules in an integrated system argument is the lack of a consistent conceptualization and terminology to describe and manage safety assurance, even within a single application domain. There is no "common approach" to safety assurance which is likely to be shared between system integrators and their supply chain, and no standardised terminology which can be used to describe and share this information in a "failproof" way. This is more than a matter of terminological inconsistency: the certification requirements of standards can vary considerably, in terms of which safety characteristics must be demonstrated and what assurance artefacts must be provided. The existence of company practices and internal standards throughout the supply chain compounds this problem.

This paper contributes the following. First we present further background information and relevant research and outline some of the essential difficulties faced when attempting to match independently-gathered assurance data from disparate sources. We assume that assurance data is constructed from a combination of evidence (design artefacts and results of analysis) and arguments (justifications as to the quality, provenance and veracity of the evidence, as well as demonstration that the requirements have been met). Assurance data is provided along with each component. We show how this can assist in the composition of component arguments, using a linguistic model of understanding to highlight matches and mismatches, ultimately helping produce a compelling system safety case. Finally, we illustrate the approach using a simple example.

II. Related Work

A. Compositional Certification - Theory

As noted in the introduction, there are numerous safety standards, guidance and best practice documents which can be used by software safety engineers to assist in the V&V process. Validation of a software component means showing that it does the task needed, typically this means showing that its requirements are correct for the system in which it is running. Verification of the software means showing that it meets those

requirements. In a design process using pre-existing or OTS components, we cannot immediately perform the validation part of the process. Instead, a piece of software is developed to a specification, and verified against it. The validation, or correctness, of that specification is only determined for a specific context or design. Other components may need to be altered or configured to work around certain weaknesses, or to accommodate mismatches in functionality.

The V&V output produced by following standards leads to the production of assurance data, consisting of evidence artefacts, e.g. design documents, test results etc. and assurance arguments about, and supported by, those evidence artefacts. For example, a safety argument about some software test results may talk about the degree of code coverage achieved and whether this is adequate to show certain properties.

Part of a compositional approach to certification requires that we take assurance data relating to discrete components and attempt to match them together. This paper concentrates on composition of the arguments. As with the design process, this has two aspects. Firstly, assessing whether evidence artefacts and, claims about it, are compatible between components (i.e. checking that the 'horizontal' claims between two arguments for composition are compatible). Secondly, ensuring that the components' function is that required to ensure system safety (i.e. checking that the claims made in the composition of the arguments can be integrated 'vertically' into the system safety argument). Further, do we have confidence that components will interact together as needed, and do we have confidence that those interactions will contribute to system safety? For example, suppose that an OTS software library guarantees to detect a particular failure mode, provided the underlying communications software responds within 6ms. Composition of the assurance arguments about both these components requires that

1. We determine if the communications software developers have made such a claim, and what the conditions are (e.g. network load)
2. We determine how much confidence we have in that evidence supporting the claim (e.g. the claim is evinced via worst case execution time (WCET) testing using statistical methods may have been performed a certain number of times under different stress conditions)
3. Whether the combination of arguments is adequate to meet any system level safety requirements (e.g. the severity of a failure to detect)

The difficulty in performing composition is that assurance arguments are written using informal natural language, and at different levels of abstraction. Therefore, it can be very difficult to match claims made about one component to those made about another one. Alternatively, claims which superficially appear to match may, in fact, differ when contextual information is compared. This paper argues that we can use principles of natural language communication in order to assist both areas of matching.

B. Related Research and Design Paradigms

There is limited research that directly addresses this problem. Therefore, in this section we mainly discuss some of the design paradigms which have been developed to assist compositional design. Further relevant research is described inline as required.

Integrated Modular Avionics (IMA) is the term used to describe a heterogeneous computing network within the avionics domain. It uses three-layer computer architecture, separating applications (system, e.g. aircraft-specific but not hardware specific), from an operating system (hardware and platform neutral - shielding the applications from any hardware changes), and a hardware interface layer (hardware specific but not system specific). One purpose of this architecture is to streamline the upgrade and maintenance of both the applications and the underlying computing hardware. Any update to software or hardware must be accompanied by an update to assurance data. Using a suitable modular certification process can assist this, limiting the amount of re-work and re-analysis as much as possible.

Research and guidance relating to how to achieve modular analysis and modular certification practically can be found in [1][2][3]. In [1] the authors looked at modular arguments, and noted that it was important to identify change scenarios in advance of certification, and also the need to look at dependency-guarantee relationships (DGRs) between assurance arguments to look at inter-operability. The authors did not develop a method to assist in the definition and expression of the DGRs. In [2] failure mode analyses of an IMA operating system were performed, and a number of challenges found, including the need match informal analysis results of the OS, and to determine whether assumptions made about how the OS was being used were correct. The avionics development guidance document DO-297 [3] defines a high-level process for certifying IMA, again requiring all assumptions to be stated when components are analysed independently, and a staged process to integration to help ensure these assumptions are as correct and complete as needed.

The newly released automotive standard in the automotive domain ISO26262 [4] has the concept of a Safety Element out of Context (SEooC). This allows a developer to produce a component, and analyse it, outside of a specific system development. The SEooC is developed to a set of assumed requirements, and makes assumptions about where and how it will be used in an actual system. The assumptions must be documented, and can then be validated during an actual vehicle development.

This common theme of the complexity in composing and integrating assurance data produced from different sources motivates our research into how to identify mismatches, gaps and matches in the assurance argument.

In [5] the authors present a restricted language for describing rely/guarantee conditions between software applications and computing hardware. This has the advantage that it can be used to generate automatically a limited set of arguments about the composed behaviour of the software, including for failure behaviour. However, it does not capture

additional information, such as relating the evidence supporting rely/guarantee claims or our confidence in them. In addition, it is limited to one specific set of software relationships to do with resource management. Arguments would also need to be generated about the validity of the rely/guarantee conditions in the argument. Our approach will cover these further aspects, but not generate the arguments automatically. Instead, we can present a list of mismatches in claims and evidence (which may be identified using some machine checking) for further examination and resolution.

III. Nuanced Conceptual/Terminological Matching

Mismatches in terminology and concepts used in the component-level justification can contribute to some of the challenges for compositional safety arguments described above: for example, evidence artefacts and justification presented in different modules might be expressed at different levels of abstraction, leading to leaps of logic in the decomposition of the argument, particularly in terms of scope of claims made and the context within which they are assumed to be valid. One possible mitigation for these problems is the use of a controlled vocabulary, which is used to formulate the key claims in the integrated argument and to reduce mismatches in the component argument modules.

Previous approaches to the use of controlled language in the requirements engineering domain [6][7] have led to the development of well-focussed, stratified requirements documents within projects and promoted greater understanding and common purpose between stakeholders. However, there are some barriers to their likely adoption for use in the development of compositional safety arguments for safety-critical, software-intensive systems. These include the difficulty in agreeing a standardised lexicon given the variety in terminology and conceptualization evident in the standards and domain practices, and the impracticality of imposing common terminology on a protracted and disparate supply chain, especially one where components are essentially OTS and are not necessarily developed for exclusive use in a single domain (or even necessarily primarily for safety-critical use at all). There is also likely to be a human factors element, in that it is difficult to mandate language use on writers.

Rather than imposing the use of a strict vocabulary on argument module developers, we propose a more nuanced approach. Component-level argument modules are developed independently, by engineers using uncontrolled natural language. Before these modules are integrated into the overall safety argument, language usage in the component-level modules is checked for mismatches, to ensure that module composition is possible. We aim to develop tool support for this checking, and to provide guidance as to the nature of the mismatches found and 'gaps' which must be filled to ensure whether the modules can be composed meaningfully. The checks and guidance cover both the language used in the argument claims, and that used to characterize the evidence artefacts used to support the arguments, so that the degree of match with what is expected in the overall argument is made clear.

Our approach requires a more subtle understanding of the relationship between terms and concepts than the *is-a* hierarchy provided by a conventional ontology. The theoretical underpinning of our approach can be explained using Saussure's paradigm of natural language communication [8]. An act of communication can be understood as a mapping relationship between a source language and a target language. A central concept in Structuralist Linguistics [8] is that of the *sign*, a two-part relation by which a term (a word or a phrase), the signifier, is associated with the conceptual entity it denotes, the signified. The sign relation is captured in Figure 1:

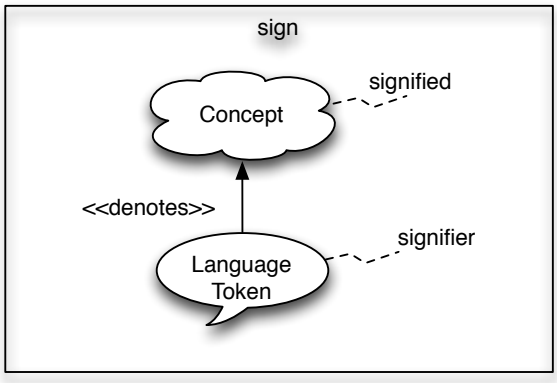

Fig 1: Sign = Signifier + Signified (after Saussure [8])

In the simplest communication scenario – where the hearer's understanding of the words used corresponds exactly to the speaker's intenntion in using them -, the match between the concept invoked in the source language (as used by the speaker) and that received in the target language (as reconstructed by the hearer) is complete. In linguistics, this relationship is called a *synonymy*. Figure 2 represents the lexical match in terms of the shared concept:

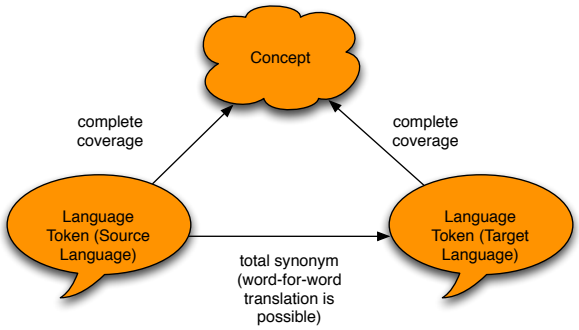

Fig 2: Complete Lexical Match (Synonymy)

However, as Lyons [9] explains, communication is rarely so straightforward. Every individual uses terms coloured by a highly individualized penumbra of contextual information, which informs the way he understands and uses a given signifier (his 'idiolect'). Various kinds of mismatch can occur:

- Homonymy: a given signifier may relate to more than one signified, i.e. a term may have more than one permissible meaning. The hearer is required to disambiguate the speaker's intention, usually by referring to contextual factors. This kind of ambiguity is bi-directional: the source language and the target

language may each use the same term but be indicating different concepts.

- The speaker may use a term that captures some aspect of the concept, while the hearer interprets by substituting a term that captures part of the same aspect of the same concept (i.e. a partial synonymy). In particularly difficult cases, the hearer's term might also introduce some other aspect of the concept. Neither term provides complete coverage of the shared concept, and there is an incomplete match between them.
- The speaker may use a term which denotes a concept which has no parallel in the hearer's idiolect: i.e. there is a signifier in the source language which is associated with a signified that has no signifier in the target language.

In cases where there is no conceptual equivalence between the languages, no matching is possible at the same level of detail. However, a partial match may be made to a "superconcept". A superconcept is a generalization of the concept, which provides a superset of its possible meanings, to which the concept in the source language can be matched, and which has some resonance in the target language. This kind of matching is depicted in Figure 3. Communication through a superconcept is inherently weak, since the precision of the concept which the speaker wished to convey is necessarily diluted by the hearer's need to conceive it through a more abstract concept. This kind of mapping is at the heart of a Thesaurus model of language, where words are grouped together according to their relationship to a core concept, of which they each capture narrow shades of meaning [10].

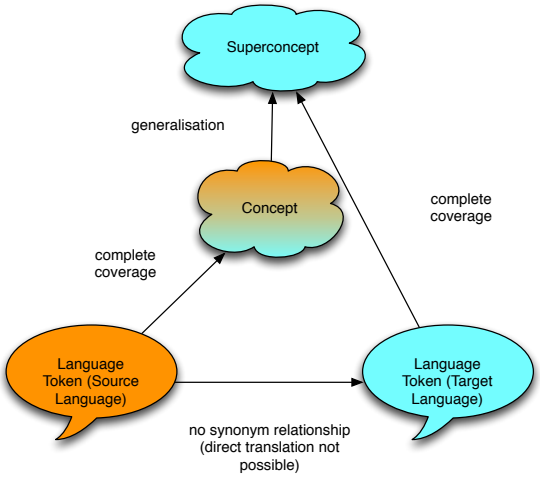

Fig 3: Weak Communication through a "Superconcept"

Rather than attempting to provide a unified vocabulary, into which component argument modules are 'translated', our approach centres around the development of a thesaurus-style vocabulary (currently in early stages of development), where the individual terms used in the standards are associated with core certification concepts. Although it is anticipated that there will be a considerable 'diplomatic' effort involved in securing cross-domain agreement on the definitions of the core concepts, there is a considerable 'buy-back' opportunity here. The conceptual vocabulary should result in a reusable 'safety-critical domain dictionary' to which other domains and projects could link.

Since component-level argument modules are developed according to the norms of a particular regulatory context or industrial domain, the "free" natural language used to express them will reflect these norms. The thesaurus is structured according to varying degrees in the matches between the terms used in the standards and the core concepts defined in the vocabulary, mirroring the lexical relationship types discussed above:

- *Exact match* – the relationship between a term from a standard and the vocabulary's term for the core concept is a synonymy.
- *Partial or fuzzy match* – some aspect of the core concept in the vocabulary is covered by a standard-specific term, but the relationship is not a synonymy.
- *No match* – a standard-specific term cannot be matched exactly to the vocabulary term for the core concept, but a match via a more abstract superconcept might be possible.

Once the thesurus definitions have been fixed and the matching of standard-specific terms to them has been established using the types described above, we use them to examine the language used in component-level argument modules being considered for composition. We run two types of pre-check: a 'horizontal' pairwise check across candidate modules for composition at the component level of the argument (largely for verification purposes), and a 'vertical' check for modules to be integrated into the overall system-level argument (the validity of requirements). The 'horizontal' check is a pairwise comparison of the nouns, adjectives and verbs used in claims between two peer modules. We check for similarity of the subjects of the claims, to identify whether the modules are talking about compatible concepts, at the same level of abstraction, or whether reference to a superconcept in one of the modules suggests a difference in the level of detail and a weakness in the composition due to potential mismatches of detail. At this stage, we also check the nouns used in the contextual statements and assumptions made in the argument modules, to identify any gaps or inconsistencies in the assumptions. A check is also made against qualifiers, to identify potential mismatches between the claims, for example whether a claim of "fault-free" operation is compatible with one about an "absence of faults". The 'vertical' check is made against the expectations of the system-level argument, to inform the integration of the module(s) into the overall argument. We check nouns, adjectives and verbs used to describe attributes of the evidence for compatibility with the expected features mandated by the template argument structure, to ensure that the overall logical structure of the system-level argument is not compromised by integration of the candidate module. Our aim is not to 'resolve' the discrepancies between terms or to provide a 'yes-no' answer or a quantitative assessment of the 'composability' or 'integratability' of the candidate module(s). Instead, we aim to provide the argument developer with intelligent feedback, based on the completeness

of the terminological matches between the modules. For example, where a fuzzy match is detected between concepts used in two component-level modules, we would provide guidance on the mismatch between the levels of detail between the modules. Where a fuzzy match is detected between a module for integration and the template argument, the system would provide guidance on likely shortcomings of the evidence and argumentation offered by the component-module, with respect to the expectations stated for the evidence in the template.

IV. Assurance Argument Example

For illustration purposes we present a small example in this section, using the Goal Structuring Notation (GSN) with modular extensions, allowing for re-usable and composable argument structures [11]. This illustrates the concepts in the previous section, and shows how the matching process can be used to assist in both bringing claims together as well as helping with gap analysis.

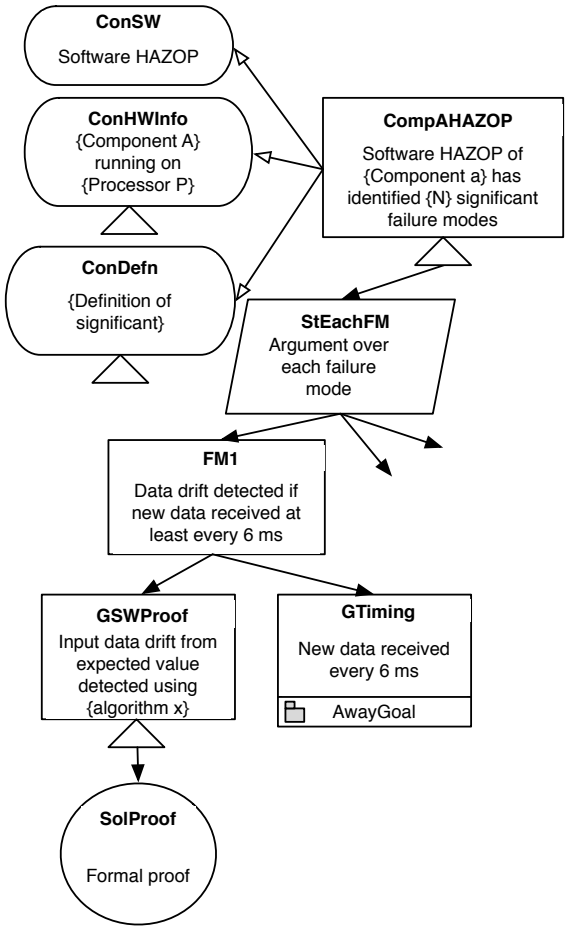

Fig 4: Software Component Failure Analysis Argument

In figure 4 an example component level assurance argument for a piece of software **{Component A}** is presented using the GSN. In the diagram, the rectangular boxes represent claims made about the software (also called goals), the lozenge shapes provide contextual information, and the parallelogram represents the strategy used to break down high-level claims into smaller claims. The triangles under some items indicate that textual information within curly braces needs to be instantiated within that claim (e.g. **{Component A}** for the name or identifier of the piece of software that was analysed). Circles are solutions, which refer to specific pieces of evidence (in this case a formal proof). This argument fragment is incomplete and has been highly condensed, but is provided to illustrate the sorts of claims that may be stated for software, and the sorts of assumptions that may need to be made about other pieces of software. The main thrust of the argument is that a software HAZard and OPerability analysis has been performed to identify significant potential failure modes within the software (significant may mean that it can no longer guarantee correct or timely output - whether this is a significant safety risk can only be assessed for each specific system i.e. it is a vertical issue). For one identified failure mode, a drift from the expected input value can only be detected if data is received at least every 6 ms. The latter claim (**GTiming**) has to be an assumption about the performance of the underlying computing platform, and cannot be guaranteed by the manufacturer of **{Component A}**. This is indicated by the grey "Away Goal" symbol at the bottom.

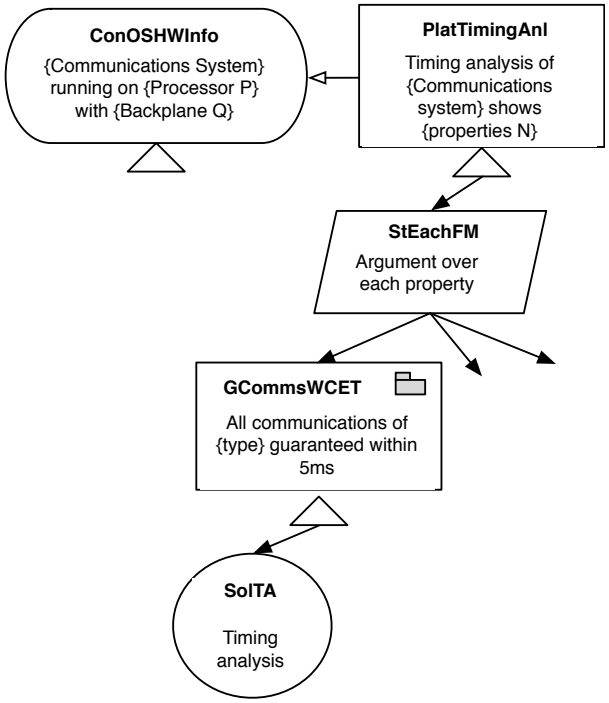

Fig 5: Supporting computing infrastructure argument

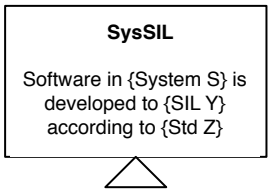

Fig 6: System level argument

Figure 5 shows an example fragment of the assurance argument for the generic computer platform on which the software might run. This argument concentrates on the timing

properties of the system, specifically that of the communications system. Again, the argument requires some items to be instantiated. The goal **GCommsWCET** provides a guarantee that communications of a specific type will be completed within 5ms. This is provided as a "Public Goal" which may be used by other parts of the argument to fulfill an assumption. This should ideally map to the away goal **GTiming**, but a superficial reading shows that the goals do not immediately align. Finally, in figure 6, we show a system specific goal, showing a typical claim about development to a specific Safety Integrity Level (SIL).

Vertical matches are required between **SysSIL**, **CompAHAZOP**, and **PlatTimingAnl** to show that component level assumptions/requirements are valid for the specific system. Horizontal matches are required between **GTiming** and **GCommsWCET** to compose the two argument modules. The following table discusses how the various types of matching are useful when comparing claims. Our ultimate aim is to assist this matching process with software tool support, identifying as many different types of match as possible. Once we have identified the matches, mismatches, missing information and partial matches we can construct a further argument (known as a contract within GSN) which describes how we should resolve any differences and if they potentially affect system safety, or (more subtly) our ability to demonstrate system safety. Thus the matching process also helps in argument gap analysis, and resolution.

TABLE 1: NUANCED MATCHING OF CLAIMS AND ARGUMENT ELEMENTS

Goals	Comments
GTiming, GCommsWCET	These goals both refer to the communications, and there is a match in the time taken for data to arrive. Although the values are different, 5ms is within the tolerances specified for **{Component A}** therefore we have an exact match. However, this is dependent on whether the communications **{type}** used by **{Component A}** is the correct one. The latter information is missing and must be provided or identified.
ConHWInfo, ConOSHWInfo	Contextual information in a GSN argument captures vital background information within which the argument is valid. Comparing the two contexts given here we see that there is potentially an exact match in the type of processor being used, however in **ConOSHWInfo** additional information is given which is simply missing from **ConHWInfo**. The person composing the two modules must assess whether this missing information is relevant or weakens the overall argument.
SysSIL, CompAHAZOP, PlatTimingAnl	Matching these two claims presents an example where a superconcept is useful - suppose **{SIL Y}** in **{Std Z}** requires that a Failure Modes and Effects Analysis is performed on all components. Software HAZOP is a specific type of FMEA, which cannot be exactly matched. Similarly, Timing Analysis may be called by a different name, such as schedulability analysis, or be part of general performance characteristics.

Two further considerations when matching, that are not explicitly demonstrated in this example, are:

- Evidence Characteristics - e.g. coverage and depth of those analyses (for the developer to work out - can we identify terms which help us?)
- Different abstraction levels (e.g. grouping the timing analysis all together)

V. Conclusions and Further Work

This paper has described some of the difficulties faced when attempting to compose assurance data, particularly arguments, which has been developed independently by different vendors. We have described how basic principles of natural language communication can be used to identify different relationships between terms and vocabulary, and how these apply in the domain of assurance arguments. Using a simple example, we illustrated how these principles appear in practice, and, further, that by performing term matching we can assist in composition of assurance arguments.

Future work will concentrate on developing the vocabulary needed for the matching process, and producing tools which can be use to automate the matching. We will explore existing technologies such as Semantics of Business Vocabularies and Rules (SBVR) and ontologies and see how we can extend them to offer sufficient flexibility in nuanced term-matching.

Acknowledgment

This work was carried out as part of the OPENCOSS Project, No: 289011, funded by the European Commission under FP7-ICT (http://www.opencoss-project.eu).

References

[1] J. Fenn, R. Hawkins, P. Williams, T. Kelly, M. Banners, & Y. Oakshott. "The Who, Where, How, Why and When of Modular and Incremental Certification", 2nd IET International Conference on System Safety. 2007.

[2] P. Conmy, "Safety Analysis of Computer Resource Management Software", PhD. YCST 2006/07, University of York, 2006.

[3] RTCA, EUROCAE, "DO-297: Integrated Modular Avionics (IMA). Design Guidance and Certification Considerations. RTCA, EUROCAE, 2007.

[4] International Organization for Standardization, "ISO26262 - Road vehicles - functional safety", November 2011.

[5] B. Zimmer, S. Bürklen, M. Knoop, J. Höfflinger, M. Trapp, "Vertical Safety Interfaces - Improving the Efficiency of Modular Certification", SAFECOMP, 2011, pp 29-42.

[6] N. E. Fuchs, U. Schwertel, and R. Schwitter, "Attempt to Controlled English – Not Just Another Logic Specification Language", LNCS vol 1559, 1999, pp 1-20

[7] A. J. Vickers, "The ConCERT Approach to Requirements Specification", Version 2, Rolls-Royce University Technology Centre in Systems and Software Engineering Report YUTC/TR/96.1, University of York, 1996.

[8] F. de Saussure, "The nature of the linguistic sign", from Cours de linguistique generale, ed C. Bally and A. Sechehaye (1916), Translated by R. Harris (1983) and reprinted in D. Lodge, Modern Criticism and Theory: A Reader, 1988, pp 10-14.

[9] J. Lyons, "Semantics", vol. 1,1977.

[10] S. M. Lloyd (ed), "Roget's Thesaurus of English Words and Phrases", 1984.

[11] GSN Community Standard, Version 1, November 2011. http://www.goalstructuringnotation.info