Improving Hazard Classification through the Reuse of Descriptive Arguments

Shamus P. Smith and Michael D. Harrison

The Dependability Interdisciplinary Research Collaboration,
Department of Computer Science, University of York,
York YO10 5DD, United Kingdom
{Shamus.Smith, Michael.Harrison}@cs.york.ac.uk

Abstract. Descriptive arguments are an intrinsic part of the process of determining the dependability of any system, particularly in the case of safety critical systems. For such systems, safety cases are constructed to demonstrate that a system meets dependability requirements. This process includes the application of hazard analysis techniques. However, such techniques are error-prone, time consuming and apply "ad hoc" reuse. Hence, the use of systematic, exhaustive hazard analysis can lead to an illusion of high confidence in the parent dependability argument that is compromised by lack of rigour.
We have investigated the application of structure and reuse techniques to improve hazard classification arguments and their associated parent dependability arguments. A structure for hazard arguments has been presented and an example from a software hazard analysis has been exemplified using XML. Using two methods of structural reuse, hazard arguments can be improved for both argument generation and post argument construction analysis.

1 Introduction

Descriptive arguments[1] are an intrinsic part of the process of determining the dependability of any system. This is particularly the case in evaluating the dependability of safety critical systems. For such systems, safety cases are constructed to demonstrate that a system meets dependability requirements. These dependability requirements are typically verified against a system's specification via demonstrated proofs and arguments that support the development, implementation and testing of the system.

Part of this verification process is the use of techniques for systematic hazard analysis. Hazard identification, classification and mitigation techniques establish that either hazards can be avoided or that they will not affect the dependability of the system. To aid this process, descriptive arguments are commonly produced to mitigate, and therefore down play, the severity of hazards and the frequency of hazardous events/states.

[1] We consider descriptive arguments as informal arguments in contrast to more quantitative, numeric arguments.

C. Gacek (Ed.): ICSR-7, LNCS 2319, pp. 255–268, 2002.

However, there are two main problems with reliance on hazard based safety arguments. Firstly, there is the problem of collecting and documenting all the relevant data. Commonly there are large amounts of raw data/evidence that needs to be documented. This process can be repetitive and error prone. Inconsistency over the gathered evidence can lead to casual ("sloppy") arguments.

Secondly, hazard analysis can be a lengthy process. This is particularly the case when exhaustive and systematic methods are used to consider potentially hazardous events and states. One result of this is that the analysis may be terminated prematurely. Typically, analysts justify this in two ways. Firstly, by stating that all the relevant hazards have been identified and it is presumed that no new hazards will be found in the untested areas and secondly, by making high level reuse of already completed analysis. This reuse is applied by describing the new analysis through difference/variations of the completed analysis. This can lead to inconsistencies in the application of "ad hoc", potentially unjustified, reuse as might occur in verbatim cross-referencing of evidence components for example.

We claim that the use of systematic, exhaustive hazard analysis can lead to an illusion of high confidence in the parent dependability argument that is compromised by lack of rigour in the analysis application and the associated argument definition.

We address these issues by demonstrating how dependability arguments can be improved by the systematic reuse of descriptive argument components. We propose that argument structures can be identified and reused based on the similarity of their structure and application of use. Reuse of successful arguments can augment similar new and existing arguments. We demonstrate this method on the hazard analysis of a typical safety case.

The remainder of this paper is as follows. In Section 2 we define a simple format for argument structure. Next, argument reuse is introduced and domain data dependence is considered. Also two approaches to structural reuse are presented. In Section 4 we describe our example domain that is the use of hazard arguments as part of a safety case for an industrial expert system. In this context we have explicitly examined hazard and operability studies (HAZOP) [7] as an example of a commonly used technique for hazard analysis. Next, the application of structural reuse is described in context of our example domain. Finally, we present a brief discussion, our conclusions and scope for future work.

2 Argument Structure

A common form of argument is one that is based on a triple of (claim, argument, evidence). There is a claim about some property with evidence presented to support the claim via an argument. This structure is based on a well known "standard" argument form of Toulmin [12].

Toulmin developed a notation (described in [12]) that can be used to structure a typical argument. In its initial form, it provides a link between data (evidence), claims and warrants (support)(see Figure 1).

"We may symbolise the relation between the data and the claim in support of which they are produced by an arrow, and indicate the authority for taking the step from one to the other by writing the warrant immediately below the arrow:" [12, pg99]

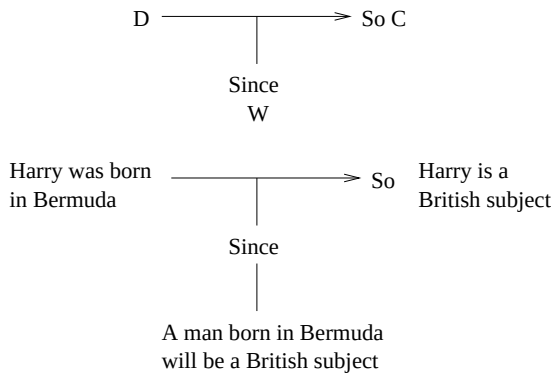

Fig. 1. Toulmin's initial argument pattern and an example

Kelly observes that Toulmin's notation can be used to express *any* argument [6, pg63]. Toulmin's arguments can be augmented with additional components, for example, qualifiers on claims and rebuttals on qualifiers, but in the context of the current work the initial definition in Figure 1 is sufficient.

What this type of argument provides is a basic structure for the definition of descriptive arguments.

3 Argument Reuse

One factor that complicates the reuse of arguments is the integration of domain specific material (e.g. data) into the argument structures. This has led to two main approaches for the reuse of arguments, one that is based on domain dependent reuse, the reuse of the data, and the other, domain independent reuse, the reuse of the argument structures and/or argument process or technique.

3.1 Domain Dependence and Independence for Reuse

For domain dependent reuse, it is the data within the domain and how it is used within an argument structure that is important. This can involve matching the current argument examples with cases in previous arguments, based on the similarity to the data/situations under discussion.

Traditional case-based reasoning (CBR) techniques can be used to determine similarity between example cases. However, this approach limits the amount of reuse that is possible. The domains under consideration would have to share

fundamental characteristics if sensible results from the reuse could be obtained. An example of this is the reuse of arguments in legal cases. Libraries of previous cases can be defined and searched to find matches to the current case in terms of outcome, legal defence, scenario etc [1]. More recently, Brüninghaus and Ashley [3] have developed a classification-based approach to find abstract situations in legal texts. Their aim is to identify indexing concepts in case texts to help the construction and maintenance of CBR systems. Components of cases are indexed via the use of simple decision trees and algorithms that utilise a legal thesaurus. Therefore the reuse of the legal cases is tightly matched to the legal domain characteristics.

Similarly, if a common theme can be identified then templates of reuse can be defined. An example of this can be seen in Kelly's use of patterns for safety cases in safety critical systems [6]. In Kelly's work the overall theme is safety critical systems and the reuse of safety case patterns over a library of example argument templates. Although Kelly describes examples of domain independent and domain dependent patterns in safety cases, all the pattern examples are firmly grounded in the overall domain of safety critical systems. The reuse is at the domain level (e.g. using patterns as a reuse technique) and not necessarily at the evidence (data) level.

It is less clear how reuse in arguments can be applied in a data independent fashion. If we cannot examine the domain data explicitly (or how it is used) to determine similarity, an alternative measure of argument similarity will need to be found. This is the focus of the work presented in this paper. We have been investigating the structure of arguments and the types of claims/evidence that can enable reuse.

3.2 Approaches to Structural Reuse

We have applied the simple argument structure of Toulmin [12], as discussed in Section 2, to define argument structures via claim $\rightarrow$ argument $\rightarrow$ evidence relations. Two data independent structures are formed by this approach. Firstly, tree like parent/child relations (for direct support) and sibling relations (for diverse support). The structures can be defined using methods based on depth first and breadth first approaches, respectively. In the next two sections, we describe how these methods can define potential for reuse.

Depth first reuse. The depth first approach involves examining the sub-tree of a particular claim and determining if it can be reused to aid support of a similar claim. For example, suppose we have claim A that has two sub-levels of support. If we introduce a claim B and can determine that claims A and B are similar, is it possible to reuse the justification structure of claim A to strengthen the argument of claim B? A pictorial example of depth first reuse can be seen in Figure 2. An individual's claim of free travel through Europe is supported by a claim of general free travel for UK passport holders **and** a claim of passport ownership. The support for the passport ownership is then reused for a second claim.

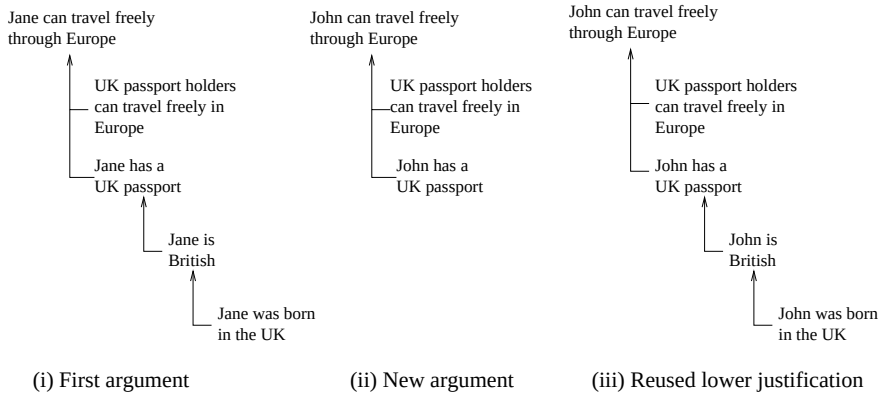

Fig. 2. Depth first reuse example

Breadth first reuse. Reuse via a breadth approach involves investigating the pairings of sibling arguments. Multiple sibling arguments at the same level can provide diversity to an argument. Diversity is desirable as independent argument strands make the overall argument more robust [6, pg154], i.e. if the strength of one child argument strand is weakened, it may not have a large effect on the overall argument/claim strength. Diversity reuse involves examining claims built on diversity arguments to determine if similar claims (i.e. the diverse structure) can be applied in other argument applications. For example suppose claim A is supported by three sub-claims (A_1, A_2 and A_3). Now we introduce claim B which is currently only supported via sub-claim A_2. The question is, can we reuse the diversity structure of the claim A argument to claim B, i.e. do A_1 and A_3 also support B? (For an example, see Figure 3.) This type of reuse can also have further implications because reuse at one level of diversity may allow reuse of that claim's sub-claims, also incorporating the depth first reuse.

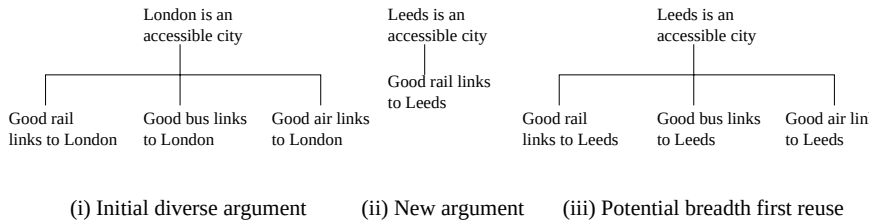

Fig. 3. Breadth first reuse example

4 The Example: DUST-EXPERT

The work that is described in this paper is grounded in a real world domain. We have been using the safety case of a software package to test/examine the reuse of arguments. The package we have been using is the DUST-EXPERT expert system tool developed by Adelard [4]. In Sections 4.1-4.3 we present brief overviews of the application, the software HAZOP component of the safety case and the arguments that are used in the HAZOP.

4.1 DUST-EXPERT

DUST-EXPERT is an application that advises on the safe design and operation of plants that are subject to dust explosions. It provides general advice on preventing dust explosions. User-extensible databases on the properties of dust and construction materials are used in techniques such as decision trees to determine dust explosion reduction strategies and to calculate methods for quantitative analysis for these strategies [4].

The documentation we are using is from the DUST-EXPERT safety case developed by Adelard [4]. They state that for safety integrity requirements, the DUST-EXPERT application is classed as SIL 2[2]. The safety case focuses on the software engineering process used. In the context of the work described in this paper we are investigating the argument usage in the hazard analysis of the DUST-EXPERT safety case and more precisely, the software HAZOP.

4.2 Software HAZOP

Pumfrey [11, pg43] observes that "HAZOP is described as a technique of *imaginative anticipation* of hazards and operation problems." Although a full description of HAZOP is outside the scope of this paper (the reader is directed to [7]) we will briefly describe the HAZOP process and highlight our area of interest, namely the use of descriptive arguments.

HAZOP is a systematic technique that attempts to consider events in a system or process exhaustively. Within a particular system domain (or scenario), items (or events) are identified and a list of guide words is applied to the items. The guide words prompt consideration of deviations to item behaviour (guide word examples include LESS VALUE, MORE VALUE, NO ACTION and LATE ACTION) to elicit the potential for arriving at possible hazardous states. These guide words provide the structure of the analysis and can help to ensure complete coverage of the possible failure modes [11, pg45].

Before starting the HAZOP, a domain description, a description of the system (normally a flow diagram), a list of elements of the flow diagram called items and a group of guide words are selected. The HAZOP process is then the exhaustive application of the guide words to each item in a particular context (using the collective knowledge a multidisciplinary team).

[2] Safety integrity level SIL 2 implies that under a low demand mode of operation, the probability of a failure for a safety function is in the range of $>= 10^{-3}$ to $< 10^{-2}$ [2].

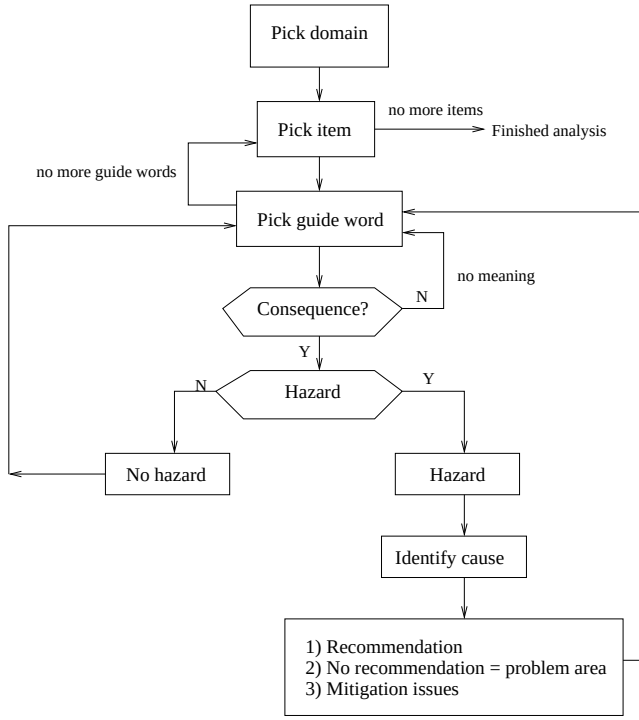

Fig. 4. Flow diagram of the HAZOP process

At each of these application steps, an implication is identified for the current item/guide word pairing. This implication has three possible results; no meaning, no hazard or hazard. This process is illustrated in Figure 4.

No meaning is used when the current guide word is not applicable for the current item. This is determined via an argument that there is no consequence for the current item/guide word pairing and such a pairing is not valid in this context (defined via expert judgement of the HAZOP team). When a *no meaning* is selected, the analysis can then move onto the next guide word.

Alternatively, a HAZOP pairing will be associated with a consequence. That consequence will either have a *no hazard* or *hazard* implication label. A *no hazard* implication is determined by there being mitigating factors (via expert judgement as understood by the HAZOP team) in the context of the consequences. These factors alter the weight of the consequences so that they fall below some threshold (also as assessed by expert judgement). For example, there may be a low impact on the system or there may a very low likelihood of it happening. Part of this process involves identifying the cause and any recommendations.

A hazard is defined when the consequence of a HAZOP pairing can not be completely mitigated. Arguments that mitigate some part of the consequence may be defined but in this case they do not change the label from *hazard* to

no hazard. This is shown in Figure 4 by three alternatives. A hazard has some recommendation (which does not mitigate the hazard), or there is no recommendation so the hazard is marked as a problem area, or the hazard has some associated mitigating arguments.

4.3 Descriptive Arguments in HAZOP

Our interest in HAZOP is concerned with the use of arguments for hazard classification. These arguments can be used to demonstrate that the hazard classification process is accurate/valid. Typically, such classification arguments are implicitly considered in the construction of the HAZOP data. However, much of this reasoning can be extracted from the data in the form of descriptive arguments. Hence, we have reinterpreted the HAZOP data in a Toulmin style.

There are two main types of argument that we have investigated. These are the *no meaning* implications and the *consequence mitigation*. As described in Section 4.2 these types of argument are used in the classification of hazards and, in the case of *consequence mitigation*, can be used to justify hazard labels and recommendations.

We have examined arguments elicited from HAZOP analysis of the DUST-EXPERT software to determine whether reusable structures can be identified. We wish to strengthen mitigation claims with previously defined "strong" arguments. However, before this analysis process could begin, the descriptive arguments were reconstructed from the raw HAZOP data.

5 Structure and Reuse

5.1 Building a Structure in the DUST-EXPERT Example

The DUST-EXPERT HAZOP information was provided by Adelard [4] in the form of a text table. This contained 330+ individual HAZOP rows. Even though this was not the complete HAZOP, this was more than enough raw data to make manual searching for patterns impractical. This was made more complicated by the fact that the structure of the data was flat. Additionally, the arguments were not clearly marked in the text and needed to be inferred from the HAZOP consequence, implication, protection and recommendation elements. The first step in managing this data was to move it to an alternative structural representation. We have done this using XML (Extensible Markup Language)[3].

5.2 Argument Structure and Reuse in XML

In line with our desire to reuse material we have built an XML structure to house the HAZOP study data and the arguments that are implied within the data.

[3] There is a vast array of texts on XML including [8].

One of our overall aims is to reuse the argument structures over different domains. Therefore we have kept the raw data separate from the argument definitions. This simplifies the searching/filtering techniques that can be used on the XML structures.

Within the XML argument structures we are using the basic support hierarchy defined by Toulmin (see Section 2). As we are investigating a data independent method, claims have been classified by *type* to allow the nature of the argument structure to be defined. Initial analysis of the raw data identified seven claims types:

- *Failure claims* are claims where mitigation is determined through the inaction of some event. For example, a help window may not appear and the support for the mitigation of this issue as a hazard is that "no action is not a hazard".
- *Duplication claims* involve the determination that multiple occurrences of an issue is allowable. For example if multiple identical help screens appear in some context where "redundancy is not a hazard". Therefore the potential for redundancy to be identified as a hazard has been mitigated by this claim.
- *Testing claims* indicate that a particular consequence is not an issue (e.g. can be mitigated) as test cases can be used to determine that the fault has not been implemented.
- *User claims* involve the participation of the user in context to defend against a consequence. For example some consequence may be "obvious to the user" and hence unlikely to happen and/or corrected quickly.
- *Feedback claims* indicate that specific information has been provided to the user to allow them to avoid a hazardous situation. For example, input values may be stored in a log file and the user may be explicitly prompted to verify the data in the log. It is assumed that the feedback is at a level so that the user will definitely become aware of it.
- *System claims* are claims that are specific to system operations. Examples include internal validation of data, automated features (e.g. window updates) and operating system restrictions (e.g. modal dialogs).
- *Timing claims* negate a consequence by identifying temporal mitigations. For example, a window may be slow to cancel but this is determined as a *no hazard* as "slow operations are not a hazard" in this context.

However, it should be noted that these are not necessarily an exhaustive set of claim types. These are only the claims that were identified for this particular example/domain.

The XML structure can be translated into HTML (HyperText Markup Language) using filters, written in JavaScript, to traverse the XML tree structure searching for arguments that match set criteria. Example criteria could be (i) all the arguments that started with a failure claim or (ii) all the arguments that start with three diverse argument threads that include a user claim.

A simple visual representation of the XML tree structure for arguments of interest can be displayed to the user via HTML in a standard web browser

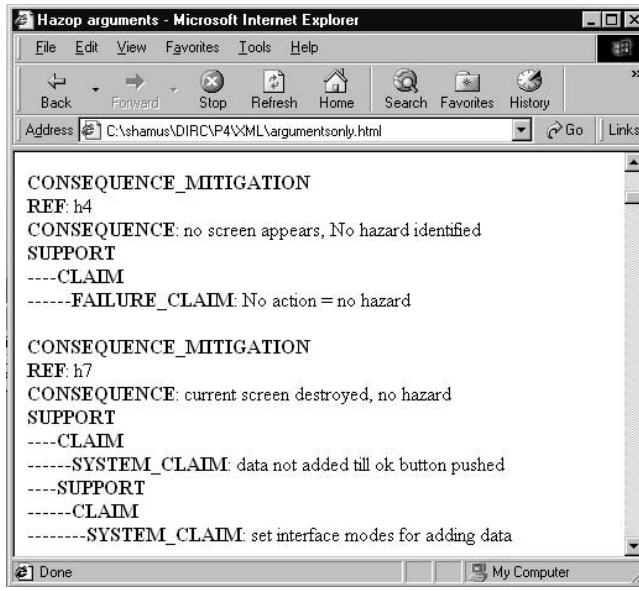

Fig. 5. XML tree representation in HTML format

(see Figure 5). It was intended that patterns of argument usage and structural similarity could then be identified for reuse purposes.

Initial experiments involved depth and breadth first filters have provided promising results on a sample of the DUST-EXPERT software HAZOP arguments.

5.3 Depth First Example

In this example, we applied a filter to the data to select all the arguments that started with a particular type of claim, e.g a duplication claim. These arguments were then examined to determine the similarity between the consequence item of the arguments and the justification of the HAZOP hazard or no hazard label. If one of the arguments has more justification (i.e. a deeper tree structure) and/or has a higher level of strength associated with it, it may be possible to reuse this arguments' justification with the other argument. This is best illustrated in a real example.

Figure 6 shows part of the output from applying the depth first filter to a subset of the HAZOP arguments. In this case, both arguments have top level duplication claims that redundancy is not a hazard. However, the second argument has further justification to this claim with diverse testing and system claims. The reuse we would be considering here would be if the same diverse pairing could be applied to the first argument in the Figure 6.

Thus we are reusing sub-trees of justification. It is hoped that this additional information will improve the overall argument mitigation.

CONSEQUENCE_MITIGATION
REF: h28
CONSEQUENCE: several identical help screens appear, no hazard
SUPPORT
----CLAIM
------DUPLICATION_CLAIM: redundancy = no hazard

CONSEQUENCE_MITIGATION
REF: h52
CONSEQUENCE: multiple records filled in - no hazard identified
SUPPORT
----CLAIM
------DUPLICATION_CLAIM: redundancy is not a hazzard
----SUPPORT
------CLAIM
--------TESTING_CLAIM: Test for identical records
----SUPPORT
------CLAIM
--------SYSTEM_CLAIM: System may not allow duplicates to be defined

Fig. 6. Example of possible reuse for a duplication claim

5.4 Breadth First Example

Using the breadth first filter, we wish to find patterns in the diverse argument threads. Ideally, there would be groupings of claims that are common over particular domains, for example if we could determine that every user claim has a sibling testing claim in each of its pairings. Therefore, we examined, over a subset of the HAZOP arguments, all the arguments that initially had two diverse claims where at least one of which started with a user claim. This generated eleven arguments of which eight comprised of user and testing diverse claims. An example of two such arguments can be seen in Figure 7.

In terms of reuse, we would be looking to augment other user claim arguments with information that supported a testing claim or to add alternative claims. For example if we had a single user argument thread, we could review other (common) testing claims to strengthen the justification for the consequence mitigation.

In this case the motivation is to strengthen the argument by adding more diverse argument threads. By identifying common threads from the existing structure, we can speed up the refinement process for arguments and also reuse the new sub-tree structures for claims which also incorporates depth first reuse.

6 Discussion

In Sections 5.3 and 5.4 we have described two approaches to descriptive argument reuse based on structural similarities. We propose that there are two main areas where such reuse will be beneficial, namely, post analysis argument refinement and "on-the-fly" argument construction.

CONSEQUENCE_MITIGATION
REF: h16
CONSEQUENCE: user types and nothing happens. No hazard provided user notices
SUPPORT
----CLAIM
------TESTING_CLAIM: can be picked up in testing
SUPPORT
-----CLAIM
-------USER_CLAIM: user might notice

CONSEQUENCE_MITIGATION
REF: h30
CONSEQUENCE: wrong help text displayed on top of correct text, hazard
SUPPORT
----CLAIM
------USER_CLAIM: may be obvious to user
SUPPORT
-----CLAIM
-------TESTING_CLAIM: test cases for implementation correctness

Fig. 7. Matching diverse argument threads using a breadth first filter

Post analysis argument refinement can be used to improve descriptive arguments. Due to the large number of arguments that may be associated with any safety case, it is unlikely that the strongest possible case will be defined on the first iteration. The methods we have discussed can be used to identify areas where argument reuse can be applied to improve the overall completed argument. Reuse in this manner would also maintain consistency between the defined arguments as similar arguments would share similar structures.

The second potential area of use would be in the construction of argument structures. As hazard arguments are being defined, previously defined arguments could be compared to see if their justification could be applied in the current situation. This "on-the-fly" analysis of the argument building process would allow reused argument components to speed up the definition process. Also consistency over the argument structure would be maintained. Common argument structures could be considered as a library of reusable parts. Expert judgement would still be required to authorise the reuse but by providing alternative candidate arguments, this process could be made systematic and semi-automated[4].

7 Conclusions and Future Work

We have investigated the development of dependability arguments and discussed the application of structure and reuse techniques to improve the resulting arguments. A basic structure for hazard arguments has been presented and an example from a software hazard analysis has been exemplified using XML to structure the data and allow the application of two methods of structural reuse.

[4] Therefore, the system provides the reuse candidates and the user makes the decision and applies the argument adaptation process.

Using these methods, arguments can be improved over both processes for argument generation and analysis after argument construction.

Currently, much of this work is at an early stage but we feel that it is a constructive initial step towards general argument reuse without the complications associated with explicit domain considerations. It is intended that the techniques we have discussed will be portable to new domains and other techniques that use argumentation.

However, one issue of concern is that bias may be incorporated into the reuse process. Patterns matching existing arguments may be given preference in the argument construction process. For example, new forms of arguments may be forced into the suggested structures when it is more appropriate that a new structure should be defined. In our approach, this issue is the responsibility of the user who applies expert judgement in the argument construction/adaptation process.

Another issue is the cost of the reuse process. There will be costs associated with both the organisation of the raw data into argument structures and the ease of the final reuse. Also there is the overhead of identifying appropriate reuse arguments. Such issues must be balanced against any proposed benefits. However, issues of cost and benefit typically require some form of measure to allow realistic predictions to be made. We are currently investigating a notion of confidence (and confidence in the worth of an argument) as such a measure to demonstrate that argument reuse will lead to improved arguments and consequently improved confidence in the arguments.

In regard to future work, we intend to continue with three main threads of research. Firstly, we will be adding more HAZOP definitions from the current example to enable us to identify more claim types and new clustering of claim structures. Also new criteria for searches using the depth and breadth first methods will be investigated.

Secondly, the current definition of structural similarity is quite high level. Therefore we intend to extend the structural approach to decompose consequences based on the stricter view of structural similarity as described by Plaza [9].

Finally, the current work is based on arguments used in hazard analysis and the use of HAZOP. We intend to investigate the reuse of the current XML structures and associated search methods/filters to alternative techniques which use descriptive arguments. Initially we will examine the arguments used in a technique for human error assessment, THEA [5,10]. THEA elicits data that can be used to construct arguments for design rationale about human reliability. It is hoped that similar argument structures and search criteria will be identified that can be considered as a top level of structural and argument reuse.

Acknowledgements

This work was supported in part by the UK EPSRC DIRC project, Grant GR/N13999. The authors are grateful to Bev Littlewood for comments on a draft of this paper, to Adelard who provided the data on DUST-EXPERT and to Corin Gurr for initial discussions on the nature of arguments.

References

1. Kevin D. Ashley and Edwina L. Rissland. A case-based approach to modelling legal expertise. *IEEE Expert*, pages 70–77, Fall 1988.
2. British Standards Institution, London, UK. *Functional safety of electrical/electronic/programmable electronic safety-related systems: Part 1: General requirements*, BS IEC 61508-1:1998 edition, 1998.
3. Stefanie Brüninghaus and Kevin D. Ashley. Towards adding knowledge to learning algorithms for indexing legal cases. In *The Seventh International Conference on Artificial Intelligence and Law*, pages 9–17, The University of Oslo, Norway, June 1999. ACM.
4. Tim Clement, Ian Cottam, Peter Froome, and Claire Jones. The development of a commercial "shrink-wrapped application" to safety integrity level 2: The DUST-EXPERT^{TM} story. In *SAFECOMP 1999*, pages 216–225, 1999.
5. Bob Fields, Michael Harrison, and Peter Wright. THEA: Human error analysis for requirements definition. Technical Report YCS-97-294, The University of York, Department of Computer Science, 1997. UK.
6. Tim P. Kelly. *Arguing Safety – A Systematic Approach to Managing Safety Cases.* PhD thesis, Department of Computer Science, The University of York, 1999.
7. Trevor Kletz. *Hazop and Hazan: Identifying and Assessing Process Industrial Hazards.* Institution of Chemical Engineers, third edition, 1992. ISBN 0-85295-285-6.
8. William J. Pardi. *XML in Action: Web Technology.* IT Professional. Microsoft Press, Redmond, Washington, 1999.
9. Enric Plaza. Cases as terms: A feature term approach to the structured representation of cases. In *First International Conference on Case-based Reasoning*, pages 265–276, 1995.
10. Steven Pocock, Michael Harrison, Peter Wright, and Paul Johnson. THEA – a technique for human error assessment early in design. In Michitaka Hirose, editor, *Human-Computer Interaction: INTERACT'01*, pages 247–254. IOS Press, 2001.
11. David. J. Pumfrey. *The Principled Design of Computer System Safety Analysis.* PhD thesis, Department of Computer Science, The University of York, 2000.
12. Stephen E. Toulmin. *The uses of arguments.* Cambridge University Press, Cambridge, 1958.