6-15-2012

A Systematic Approach to Justifying Sufficient Confidence in Software Safety Arguments

Anaheed Ayoub
University of Pennsylvania, anaheed@seas.upenn.edu

Baekgyu Kim
University of Pennsylvania, baekgyu@seas.upenn.edu

Insup Lee
University of Pennsylvania, lee@cis.upenn.edu

Oleg Sokolsky
University of Pennsylvania, sokolsky@seas.upenn.edu

Recommended Citation

Anaheed Ayoub, Baekgyu Kim, Insup Lee, and Oleg Sokolsky, "A Systematic Approach to Justifying Sufficient Confidence in Software Safety Arguments", *Lecture Notes in Computer Science: Computer Safety, Reliability, and Security* 7612, 305-316. June 2012. http://dx.doi.org/10.1007/978-3-642-33678-2_26

31st International Conference, SAFECOMP 2012, Magdeburg, Germany, September 25-28, 2012.

A Systematic Approach to Justifying Sufficient Confidence in Software Safety Arguments

Abstract

Safety arguments typically have some weaknesses. To show that the overall confidence in the safety argument is considered acceptable, it is necessary to identify the weaknesses associated with the aspects of a safety argument and supporting evidence, and manage them. Confidence arguments are built to show the existence of sufficient confidence in the developed safety arguments. In this paper, we propose an approach to systematically constructing confidence arguments and identifying the weaknesses of the software safety arguments. The proposed approach is described and illustrated with a running example.

Keywords

safety cases, confidence arguments, assurance deficits

Comments

31st International Conference, SAFECOMP 2012, Magdeburg, Germany, September 25-28, 2012.

A Systematic Approach to Justifying Sufficient Confidence in Software Safety Arguments*

Anaheed Ayoub, BaekGyu Kim, Insup Lee, Oleg Sokolsky

Computer and Information Science Department
University of Pennsylvania
{anaheed,baekgyu,lee,sokolsky}@seas.upenn.edu

Abstract. Safety arguments typically have some weaknesses. To show that the overall confidence in the safety argument is considered acceptable, it is necessary to identify the weaknesses associated with the aspects of a safety argument and supporting evidence, and manage them. Confidence arguments are built to show the existence of sufficient confidence in the developed safety arguments. In this paper, we propose an approach to systematically constructing confidence arguments and identifying the weaknesses of the software safety arguments. The proposed approach is described and illustrated with a running example.

Keywords: safety cases, confidence arguments, assurance deficits

1 Introduction

A safety case is a structured argument, supported by a body of evidence, that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given environment [17]. Although creating a structured safety argument explicitly explains how the available evidence supports the overall claim of acceptable safety, it cannot ensure that the argument itself is 'good' or the evidence is sufficient. A justification for the sufficiency of confidence in safety arguments is essential. Any gap that prohibits perfect confidence is referred to as an *assurance deficit* [11]. The argument about the assurance deficits is given in a separate argument that is named *confidence argument* [11]. A confidence argument demonstrates the existence of sufficient confidence in an element by showing that the assurance deficits related to this element have been identified and managed. Showing overall confidence in a safety argument would require that all elements of the safety argument (such as evidence or contexts) have an accompanying confidence argument.

In this paper, an approach to systematically identify the assurance deficits in software safety arguments is proposed. Software safety arguments are safety arguments that justify, based on evidence, that the software does not contribute to the system hazards. Following a systematic approach would help in effectively

* This work is supported in part by the NSF CPS grant CNS-1035715 and the NSF/FDA Scholar-in-Residence grant CNS-1042829.

identifying the assurance deficits. To show sufficient confidence in a specific element in a safety argument, a confidence argument developer first explores all concerns about the confidence in this element, and then makes claims that these concerns are addressed. If a claim cannot be supported by convincing evidence, then a deficit is identified and should be addressed. However, one cannot define a complete list for *all* concerns about all elements used in the safety arguments.

In this work, we are taking advantages of a commonality among elements used in software safety arguments. For example, tool qualification is one of the concerns for all tool-derived evidence [19]. Addressing a concern like this typically gives rise to several *derived* concerns. We collected common concerns for common elements used in software safety arguments. We call the set of derived concerns for a specific element *characteristics* of this element. We structured this collection of common concerns in what we called the *common characteristics map*. It is a map from a concern C from the characteristics set, to a set of derived concerns that need to be argued about to justify sufficient confidence in C. We also propose a *common characteristic mechanism* to construct confidence arguments and identify assurance deficits by instantiating the map to specific concerns. Any branch of the developed confidence argument not be supported by evidence indicates an assurance deficit that needs to be addressed.

The paper is organized as follows: Section 2 gives a brief background on safety cases. The related work is listed in Section 3. Section 4 explains the basic idea of the proposed approach. The common characteristics map is presented in Section 5. The common characteristics mechanism is described and illustrated with a running example in Section 6. The mechanism evaluation is given in Section 7. Finally, the paper is concluded in Section 8.

2 Safety Cases

The safety of safety-critical systems is of a great concern. Many such systems are reviewed and approved or certified by regulatory agencies. For example, medical devices sold in the United States are regulated by the U.S. Food and Drug Administration (FDA). Some of these medical devices, such as infusion pumps, cannot be commercially distributed before receiving an approval from the FDA [18]. Which means that manufacturers of such systems are expected not only to achieve safety but also to convince regulators that it has been achieved [20]. Recently, safety cases have become popular and acceptable ways for communicating ideas and information about the safety-critical systems among the system stakeholders. The manufactures submit safety cases (to present a clear, comprehensive and defensible argument supported by evidence) to the regulators to show that their products are acceptably safe to operate in the intended context [13]. There are different approaches to structure and present safety cases. The Goal Structuring Notation (GSN) [13] is one of the description techniques that have been proven to be useful for constructing safety cases. In this work, we use the GSN notation in presenting safety cases. There is often commonality among the structures of arguments used in safety cases. This commonality

motivates the definition for the concept of argument patterns [13], which is an approach to support the reuse of arguments among safety cases.

A new approach for creating clear safety cases was introduced in [11]. This new approach basically separates the major components of the safety cases into safety argument and confidence argument. A safety argument is limited to give arguments and evidence that directly target the system safety. For example, claiming why a specific hazard is sufficiently unlikely to occur and arguing this claim by testing results as evidence. A confidence argument is given separately to justify the sufficiency of confidence in this safety argument. Such as questioning about the confidence in the given testing results (e.g., is that testing exhaustive?). These two components are given explicitly and separately. They are interlinked so that justification for having sufficient confidence in individual aspects of the safety component is clear and readily available but not confused with the safety component itself. This separation reduces the size of the core safety argument. Consequently, this new structure is believed to facilitate the development and reviewing processes for safety cases.

3 Related Work

There exists a widely used method for systematically constructing safety arguments. This method is often referred to as the "Six-Step" method [12]. Although this method has been used successfully in constructing many safety arguments, it does not explicitly consider the confidence of the constructed safety arguments [10]. In [16, 19], lists of major factors that should be considered in determining the confidence in arguments are defined. Questions to be considered when determining the sufficiency of each factor are also given. We were inspired by this work and focused on one of these factors (i.e., the trustworthiness).

Argument patterns for confidence are given in [11]. Those patterns are defined based on identifying and managing the assurance deficits to show sufficient confidence in the safety argument. It is necessary to identify the assurance deficits as completely as practicable. However, it is not quite clear how to do that. This motivates us to take a step back to reasonably identify the assurance deficits. Then the list of the recognized assurance deficits can be used in instantiating the confidence pattern given in [11]. The constructed confidence arguments can be used in the appraisal process for assurance arguments (e.g., [6, 14]).

There are attempts to quantitatively measure confidence in safety cases such as [5, 7]. We believe that qualitative reasoning about the confidence existence is more consistent with the inherited subjectivity in safety cases.

4 Proposed Approach

The best practice for supporting the top-claim of safety arguments (i.e., the system is acceptably safe) is to show that the identified system hazards are adequately mitigated. We refer to this type of argument as a *contrapositive* argument, since it refutes attempts to show that the system is unsafe. To build

this argument, one should first determine what could go wrong with this system (i.e., identify the system hazards). Similarly, the top claim for a confidence argument is usually that sufficient confidence exists in an element E of the safety argument. Such a claim can be supported by a contrapositive argument showing that the identified assurance deficits associated with E are adequately mitigated [11]. Extending the analogy, one should first determine the uncertainties associated with the element (i.e., identify the assurance deficits). Following systematic approaches helps in effectively identifying system hazards [1]. We believe that following a systematic approach would also help in effectively identifying assurance deficits.

The proposed systematic approach to identifying the assurance deficits results in the construction of *positive confidence arguments*. A positive argument is a direct argument that relies on the properties of the actions taken in the development (e.g., a well-established development process has been followed, a trusted tool has been used, etc.). This distinguishes our confidence arguments from the contrapositive ones discussed above. We stress that the intent of our work is not to replace contrapositive arguments, but to aid in the identification of deficits that can then be argued over using a contrapositive argument. However, note that if no deficits are identified through the construction of a positive argument, the resulting argument can be used as the requisite confidence argument.

We propose a common characteristics map to provide guidelines for systematic construction for positive confidence arguments. Using the map, claims in the positive confidence arguments can be decomposed until every goal is supported by positive sufficient evidence. If all branches in the positive confidence arguments are supported by convinced evidence, that means all assurance deficits are mitigated. For each goal in the resulting confidence arguments that cannot be solved with sufficient evidence, an assurance deficit is identified and needs to be addressed. After identifying the assurance deficits in this way, the confidence pattern [11] can be instantiated to demonstrate that the recognized assurance deficits are managed.

5 The Common Characteristics Map

As given in [11], the overall confidence in a safety argument requires confidence arguments to be constructed for all context, all evidence and all inferences used in the safety argument. There are several factors that influence our confidence in system safety, such as appropriateness, independence, etc. In this paper, we concentrate on one of these factors, namely trustworthiness. Trustworthiness (i.e., the likelihood of freedom from errors) is a major factor that must be considered in determining the assurance of evidence and contexts. According to [11], trustworthiness is not a confidence factor for inferences. So the proposed work is used for context and evidence, but not inferences.

There are commonalities among contexts and evidences used within the software safety arguments. For example, software safety arguments are likely to cite tool-derived evidence. The tool qualification is one of the concerns for any

tool-derived evidence. Our observation is that elements used in software safety arguments can be categorized based on their common concerns. The categories commonly used in software safety arguments are illustrated below:

- *Created artifact*: e.g., a system model, a fault tree
- *Provided artifact*: e.g., system requirements, results from technical literature
- *Process results*: e.g., the formal verification results, the testing results
- *The use of a mechanism*: e.g., a particular design or verification technique
- *The use of a tool*: e.g., a specific model-checking or code-generation tool

We note that this list is not complete, but identifies categories that cover a collection of the more common elements used in software safety arguments. To show that this list is reasonable, we collected the contexts and evidences used in the argument patterns given in [2, 3, 10, 13, 19] and found that each of these elements can be classified as one of the listed categories.

Table 1. Concerns regarding the outcomes of formal verification and testing

Process results	Formal verification results	Testing results
the used technique	the used formal verification technique	the used testing technique
the used tool	the used formal verification tool	the used testing tool
expertise of the human involved in the process	expertise of the verification engineer	expertise of the tester
correctness of the involved artifacts	correctness of the system properties	correctness of the test cases
the relation among the involved artifacts	the coverage of the system requirements	the test coverage

Mapping the confidence concerns. Elements belonging to the same category have similar concerns about their trustworthiness, which need to be reasoned about in a confidence argument. Table 1 illustrates this similarity with an example that compares concerns regarding the outcomes of formal verification and testing, as examples of evidences cited by software safety arguments. The first column gives a generalization for the next two columns. For example, *the used tool* is a generalization that covers the used formal verification tool and the used testing tool. The formal verification results and the testing results can be categorized as *process results* as shown in the first row. We call this set of concerns *characteristics* of the category. Arguments over the characteristics of a category are to support sufficient confidence in the trustworthiness of elements that belong to this category. When we start addressing a particular concern C from the characteristics set, it may, in turn, give rise to a set of derived trustworthiness concerns, which correspond to the category exhibited by C. We illustrate the notions of concern, category of concerns, and characteristics of a category in Figure 1. For example, suppose we are addressing concern A1 that falls into the category A. Its derived concerns are B1 and C3, that fall into categories B and C, respectively. Moreover, every concern in A will have derived concerns in B and C. We then say that B and C are the characteristics of A. Several concrete examples of concerns and their categories are given below.

This relationship between categories of concerns based on the notion of characteristics can be captured as a map. We constructed such a map, shown in Figure 3, that relates each category to its characteristics that need to be argued about in order to justify sufficient confidence in elements that belong to this category. Nodes in the map are categories, i.e., sets of concerns with similar characteristics, where each characteristic is a derived concern, as illustrated above. Solid arrows connect each node to the nodes that represent categories of its characteristics. To address a claim about the trustworthiness in a node, we need to argue over the trustworthiness in all nodes reached by solid arrows from this node. For example, to show that a *process result* is trustworthy, argument about trustworthiness in all aspects of this process should be given, which include the use of a tool on which the process is based, the artifacts used in the process, etc. In turn, to address the claim about the trustworthiness in *the use of a tool*, we need to argue about the trustworthiness in the tool itself (*the tool* category), the person who used this tool (*the human factor* category), etc.

To show that a *created artifact* is trustworthy, argument about trustworthiness in its creation process should be given [9]. In addition, we need to argue that the process of validating the artifact with respect to its requirements is trustworthy. For example, both the artifact *creation process* and *validation results* exhibit the characteristics of the *process results* category and, for each, we should explore the derived concerns of that category. The dotted arrows are used in the map to demonstrate that the connected two nodes have the same characteristics. Note that we could eliminate dotted arrows altogether by combining together the nodes connected by dotted arrows. However, we believe that keeping them separate makes the map easier to follow and helps in map instantiation, described in Section 6.

Evaluation. The proposed common characteristics map guides to what should be argued for the confidence in the trustworthiness. We say that the map is considered reasonable if the concerns collected in the map cover at least all known concerns. As mentioned in Section 3, some existing work suggests questions to be asked and things to be considered for the trustworthiness factor. We collected the concerns and questions given in [11, 16, 19], and made sure that all these concerns and questions are covered in the common characteristics map. For example, concerns listed in [16] for trustworthiness are covered by the common characteristics map as follows:

- Was the evidence gathered in accordance with any documented standard? This concern is covered by the category *the use of a mechanism.*
- Are the evidence-gathering personnel competent? Are they certified to an appropriate standard? Have they performed the tests before? These concerns are covered by *the expertise of a person* category.
- How valid are the assumptions and simplifications that were made? This concern is covered by *applicability of the tool assumptions and limitations* and *applicability of the mechanism assumptions and limitations* categories.

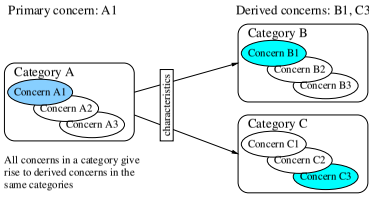

Fig. 1. Categories of concerns and their characteristics

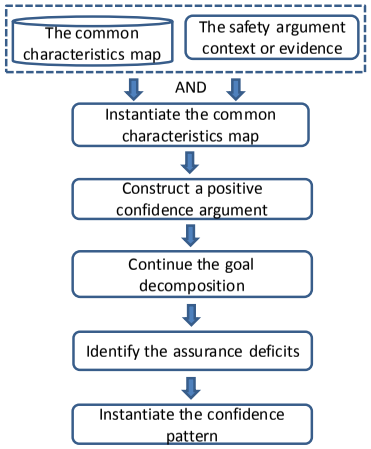

Fig. 2. The steps of the common characteristics mechanism

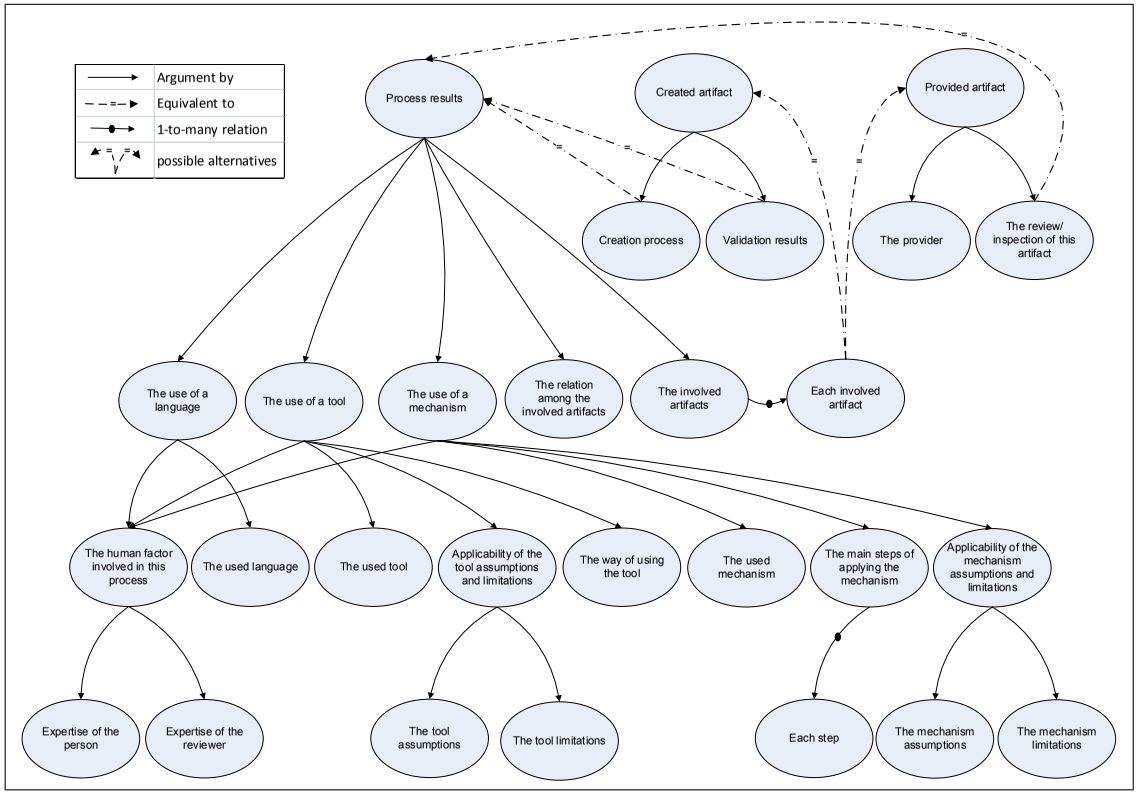

Fig. 3. The common characteristics map

6 The Common Characteristics Mechanism

The proposed mechanism starts by creating positive confidence arguments with the help of the common characteristics map. The main steps of the common characteristics mechanism are shown in Figure 2. In this section, a description for each step is given and illustrated with a running example based on a recent case study. The case study involved constructing a safety case for the implementation of a Patient Controlled Analgesic (PCA) infusion pump. The PCA infusion pump is one of those medical devices that are subject to premarket approval requirements by the FDA [18]. We developed a PCA implementation by using the model-based approach based on the Generic PCA model [8] provided by the FDA. The details of our PCA development are given in [15]. Briefly, given the GPCA Simulink/Stateflow model provided by the FDA, a UPPAAL timed automata model [4] was constructed using a manual translation process. This GPCA timed automata model is then used to synthesize the software for our PCA implementation. In [3], we have presented part of the safety argument for the resulting implementation. One of the contexts that is referenced in the PCA safety argument is the GPCA timed automata model. The context of the GPCA timed automata model is used here as a running example.

As shown in Figure 2, to construct a confidence argument for a given element of the safety argument using the common characteristics map, we first instantiate the map starting from the node in the map that corresponds to the category of this element. For example, the map instance for the GPCA timed automata model is given in Figure 4. In our example, this model falls in the *created artifact* category. We then select the corresponding node from the map and instantiate it. That is, *created artifact* node in Figure 3 is instantiated as *the GPCA timed automata model* node in Figure 4. We then unroll the map following the solid edges, and instantiate the reachable nodes: *the creation process* and *validation results*. These two nodes are instantiated to *the creation process for the GPCA timed automata model* and *validation results* nodes, respectively, in Figure 4 in the second layer. The characteristics of these nodes are the same as for the *process results* category and, in the third layer in Figure 4, we instantiate those nodes as well, and continue the instantiation process iteratively.

In the second step, we construct a positive confidence argument from the instantiated map (e.g., Figure 4). Start from the root node (e.g., *the GPCA timed automata model* node in Figure 4). Create the top-level goal claiming sufficient confidence in the trustworthiness of this element (e.g., goal G:Trustworthiness in Figure 5). For each node reached from the root node (i.e., layer 2 nodes in Figure 4), we create a strategy to decompose the top-level goal (e.g., strategies S:Trustworthy and S:Validation in Figure 5). Each node in layer 3 creates a goal in the confidence argument, and so on.

We see that the element of the confidence argument created for a node in the instantiated map depends on its layer. That is, we create goal elements for nodes in odd layers and strategy elements for nodes in even layers in the map instance. Actually the same map node can sometimes appear in even layer and sometimes appear in odd layer. For example, *process results* node is in layer 1,

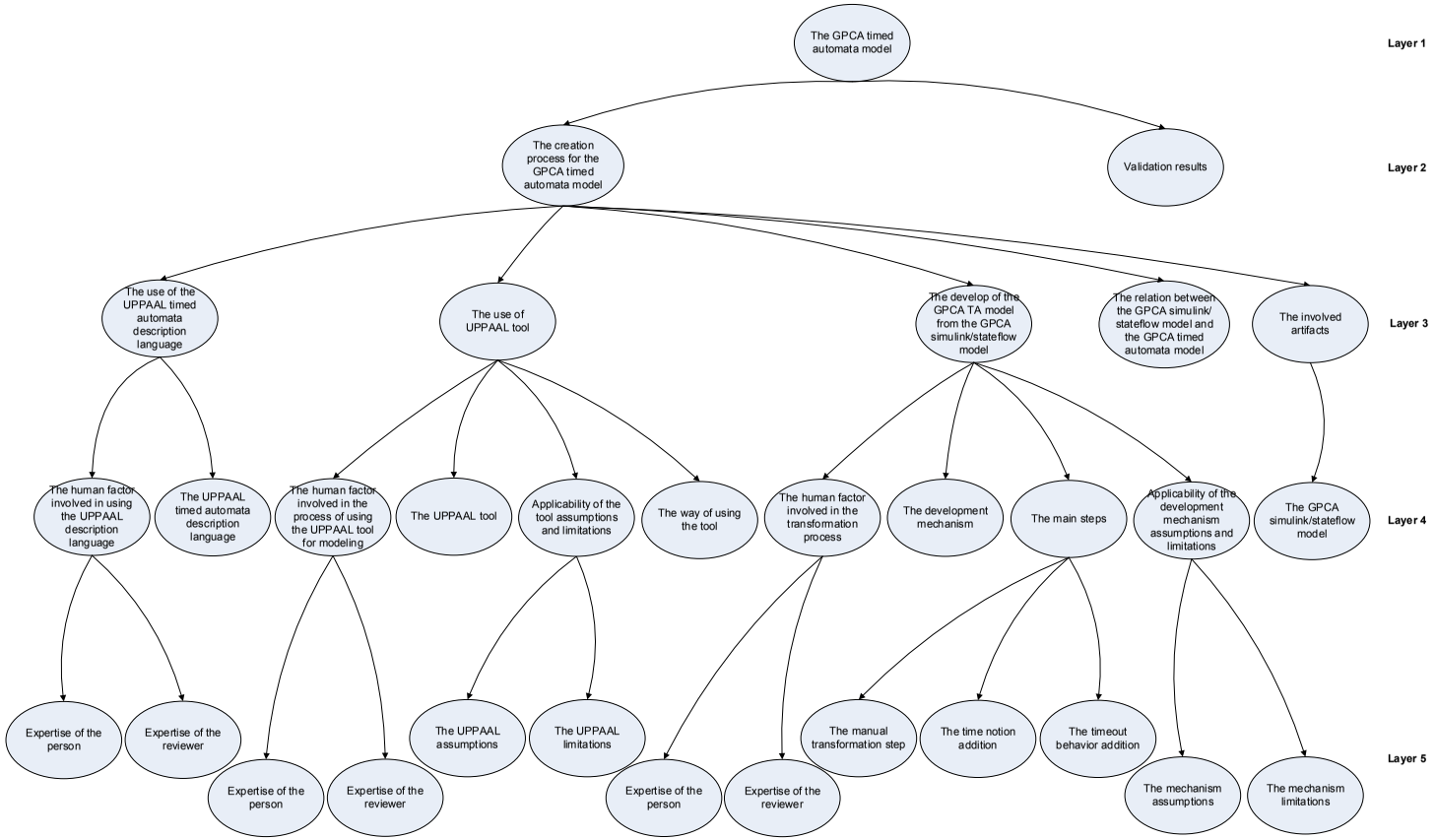

Fig. 4. The common characteristics map instance to the GPCA timed automata model

but if it is driven from *created artifact* then it will be in layer 2. In the first case, a goal will be created for it claiming the existence of sufficient confidence in the trustworthiness of the process results. In the second case, a strategy will be created with an argument by the process (e.g., argument by validation). Solid shapes and arrows in Figure 5 show part of the developed positive confidence argument for the GPCA timed automata model.

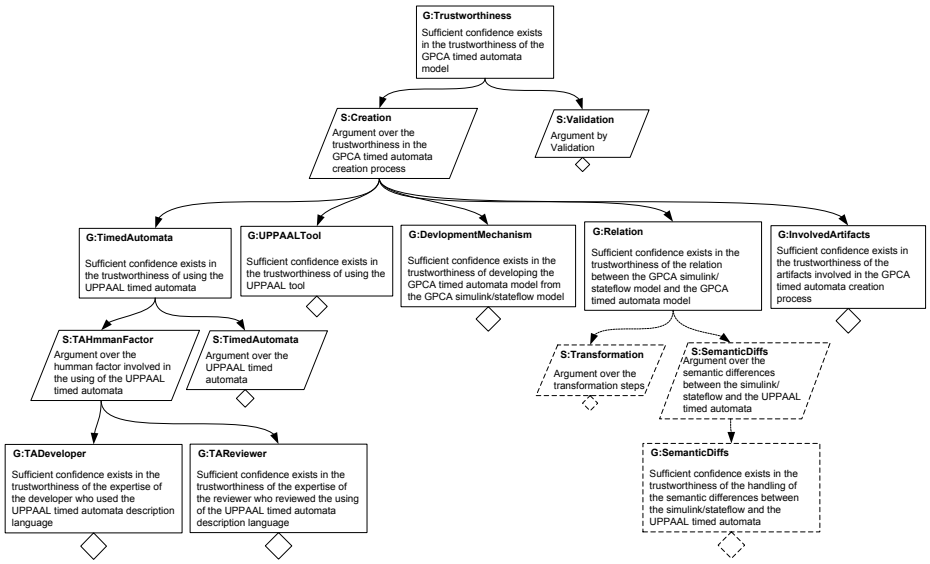

Fig. 5. Part of the positive confidence argument for the GPCA timed automata model

The decomposition for the confidence argument nodes continues until every claim is supported with evidence. The dotted shapes and arrows in Figure 5 show the elements that require further decomposition. Decomposition for G:Relation is required to support the claim about the trustworthiness in the relation between the GPCA Simulink/Stateflow model and the GPCA timed automata model. As the GPCA Simulink/Stateflow model was transformed into the GPCA timed automata model, then this decomposition is given by two strategies S:Transformation and S:SemanticDiffs. Any claim in the confidence argument that cannot be supported by evidence identifies an assurance deficit. For example, although we transformed the GPCA simulink/stateflow model into an equivalent GPCA timed automata model, we do not have evidence to show this equivalence at the semantic level. So the claim at G:SemanticDiffs is not supported and so an assurance deficit is identified here.

For the identified assurance deficits, a contrapositive argument about their mitigations needs to be constructed using the confidence pattern defined in [11]. In our case, exhaustive conformance testing between the GPCA Simulink/Stateflow model and the GPCA timed automata model may be a reasonable mitigation. We also have to instantiate the confidence argument for the trustworthiness of conformance testing from the common characteristics map.

7 Discussion

Observations. The proposed common characteristics map is not complete and so it should not be used blindly. The generated confidence arguments may require additional elements. In particular, generated goals and strategies may need contexts, assumptions, and/or justifications. For example, a justification node, stating that the GPCA timed automata model was developed from the GPCA Simulink/Stateflow model using a careful transformation process [15], should be connected to goal G:DevelopmentMechanism in Figure 5. Note that if any context or assumption is added then argument about sufficient confidence in it should be also considered.

Nodes in the map instance cannot be omitted at will during the confidence argument construction. Otherwise, confidence in the trustworthiness of the element under concern is questionable and that identifies a potential assurance deficit. For example, if tool assumptions are not known, *the tool assumptions* node indicates a weakness that should be addressed. However, not every possible derived concern has to be present. If we decide to omit a branch in the instantiation, we have to supply appropriate justification.

Limitations. The common characteristics map presented in this paper covers only the trustworthiness factor. However, similar maps can be constructed for other factors such as appropriateness. To do this, we need to identity categories of appropriateness concerns and their characteristics. We leave this as our future work. The common characteristics mechanism is not an automatic approach, i.e., it needs human interactions and decisions (e.g., what nodes can be ignored with justification and what parts should be added as mentioned above).

While the structure of the argument is directly derived from the map instance, the created goals and strategies still need to be formulated correctly. For example, goal G:TADeveloper in Figure 5 is derived from the node *expertise of the person* in Figure 4. The statement of the goal in G:TADeveloper is formed as a proposition following the rules given in [12].

8 Conclusions

It is important to identify the assurance deficits and manage them to show sufficient confidence in the safety argument. In this paper, we propose an approach to systematically construct confidence arguments and identify the assurance deficits in software safety arguments. Although the proposed mechanism does not guarantee to identify all assurance deficits, it helps to identify deficits that may have been overlooked otherwise. Similarly, following systematic hazard identification mechanisms does not guarantee that all hazards are identified.

The paper focuses on constructing positive confidence arguments with the help of a proposed map. However, the map can also be used in the reviewing process to help regulators identify gaps in submitted confidence arguments.

Our preliminary experience of applying the proposed approach has revealed that the common characteristics mechanism yields the expected benefits in exploring important uncovered assurance deficits in software safety arguments.

References

1. Federal Aviation Administration. FAA System Safety Handbook, Chapter 8: Safety Analysis/Hazard Analysis Tasks . *System*, 40(4), 2000.
2. R. Alexander, T. Kelly, Z. Kurd, and J. Mcdermid. Safety Cases for Advanced Control Software: Safety Case Patterns. Technical report, University of York, 2007.
3. A. Ayoub, B. Kim, I. Lee, and O. Sokolsky. A Safety Case Pattern for Model-Based Development Approach. In *NFM2012*, pages 223–243, Virginia, USA, 2012.
4. G. Behrmann, A. David, and K. Larsen. A tutorial on UPPAAL. In *Formal Methods for the Design of Real-Time Systems*, LNCS, pages 200–237, 2004.
5. R. Bloomfield, B. Littlewood, and D. Wright. Confidence: Its Role in Dependability Cases for Risk Assessment. In *Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on*, pages 338 –346, 2007.
6. L. Cyra and J. Górski. Expert Assessment of Arguments: A Method and Its Experimental Evaluation. In *Computer Safety, Reliability, and Security, 27th International Conference, SAFECOMP*, 2008.
7. E. Denney, G. Pai, and I. Habli. Towards Measurement of Confidence in Safety Cases. In *International Symposium on Empirical Software Engineering and Measurement (ESEM'11)*, Washington, DC, USA, 2011. IEEE Computer Society.
8. The Generic Patient Controlled Analgesia Pump Model. http://rtg.cis.upenn.edu/gip.php3.
9. I. Habli and T. Kelly. Achieving Integrated Process and Product Safety Arguments. In *the 15th Safety Critical Systems Symposium (SSS'07)*. Springer, 2007.
10. R. Hawkins and T. Kelly. Software Safety Assurance – What Is Sufficient? In *4th IET International Conference of System Safety*, 2009.
11. R. Hawkins, T. Kelly, J. Knight, and P. Graydon. A New Approach to creating Clear Safety Arguments. In *19th Safety Critical Systems Symposium (SSS'11)*, pages 3–23. Springer London, 2011.
12. T. Kelly. A six-step Method for Developing Arguments in the Goal Structuring Notation (GSN). Technical report, York Software Engineering, UK, 1998.
13. T. Kelly. *Arguing safety – a systematic approach to managing safety cases.* PhD thesis, Department of Computer Science, University of York, 1998.
14. T. Kelly. Reviewing Assurance Arguments – A Step-by-Step Approach. In *Workshop on Assurance Cases for Security - The Metrics Challenge, Dependable Systems and Networks (DSN)*, 2007.
15. B. Kim, A. Ayoub, O. Sokolsky, P. Jones, Y. Zhang, R. Jetley, and I. Lee. Safety-Assured Development of the GPCA Infusion Pump Software. In *EMSOFT*, pages 155–164, Taipei, Taiwan, 2011.
16. C. Menon, R. Hawkins, and J. McDermid. Defence standard 00-56 issue 4: Towards evidence-based safety standards. In *Safety-Critical Systems: Problems, Process and Practice*, pages 223–243. Springer London, 2009.
17. Ministry of Defence (MoD) UK. *Defence Stanandard 00-56 Issue 4: Safety Management Requirements for Defence Systems*, 2007.
18. U.S. Food and Drug Administration, Center for Devices and Radiological Health. *Guidance for Industry and FDA Staff - Total Product Life Cycle: Infusion Pump - Premarket Notification [510(k)] Submissions*, April 2010.
19. R. Weaver. *The Safety of Software - Constructing and Assuring Arguments.* PhD thesis, Department of Computer Science, University of York, 2003.
20. F. Ye. *Contract-based justification for COTS component within safety-critical applications.* PhD thesis, Department of Computer Science, University of York, 2005.