2. Software Safety Argument Pattern Catalogue Structure

The following argument patterns are provided in the software safety argument pattern catalogue:

High-level software safety argument pattern (Section 3.1) – This pattern provides the high-level structure for a software safety argument. The pattern can be used to create the high level structure of a software safety argument either as a stand-alone argument or as part of a broader system safety argument.

Software contribution safety argument pattern (Section 3.2) - This pattern provides the structure for an argument that the contributions made by software to system hazards are acceptably managed. This pattern is based upon a generic 'tiered' development model in order to make it generally applicable to a broad range of development processes and technologies.

Software Safety Requirements identification pattern (Section 3.3) - This pattern provides the structure for an argument that software safety requirements (SSRs) are correct and appropriate for each tier of the software design.

Hazardous contribution software safety argument pattern (Section 3.4) – This pattern provides the structure for an argument that hazardous errors are not introduced at each tier of software design decomposition. This includes arguing that mistakes have not been made in decomposing the design, and also that no *new* hazardous behaviour has been introduced.

Software contribution safety argument pattern with grouping (Section 3.5) - This pattern is an extension of the Software Contribution Safety Argument Pattern. It provides the option of grouping the argument to reflect natural requirements groupings in the software design.

When instantiated for the target system, these patterns link together to form a single software safety argument for the software.

The argument patterns are documented using the pattern extensions to the Goal Structuring Notation (GSN), described in Appendix A.

3. The Software Safety Argument Pattern Catalogue

3.1. High-Level Software Safety Argument Pattern

High-Level Software Safety Argument Pattern			
Author	Richard Hawkins		
Created	09/12/08	**Last modified**	08/06/09

INTENT

This pattern provides the high-level structure for a software safety argument. The pattern can either be used to create the high level structure of a 'stand alone' software safety argument considering just the software aspects of the system, or alternatively can be used to support claims relating to software aspects within a broader system safety argument.

STRUCTURE

The structure of this argument pattern is shown in Figure 1. Note that there are a number of different possible top goals for this pattern, as indicated by the public goals in the argument structure below.

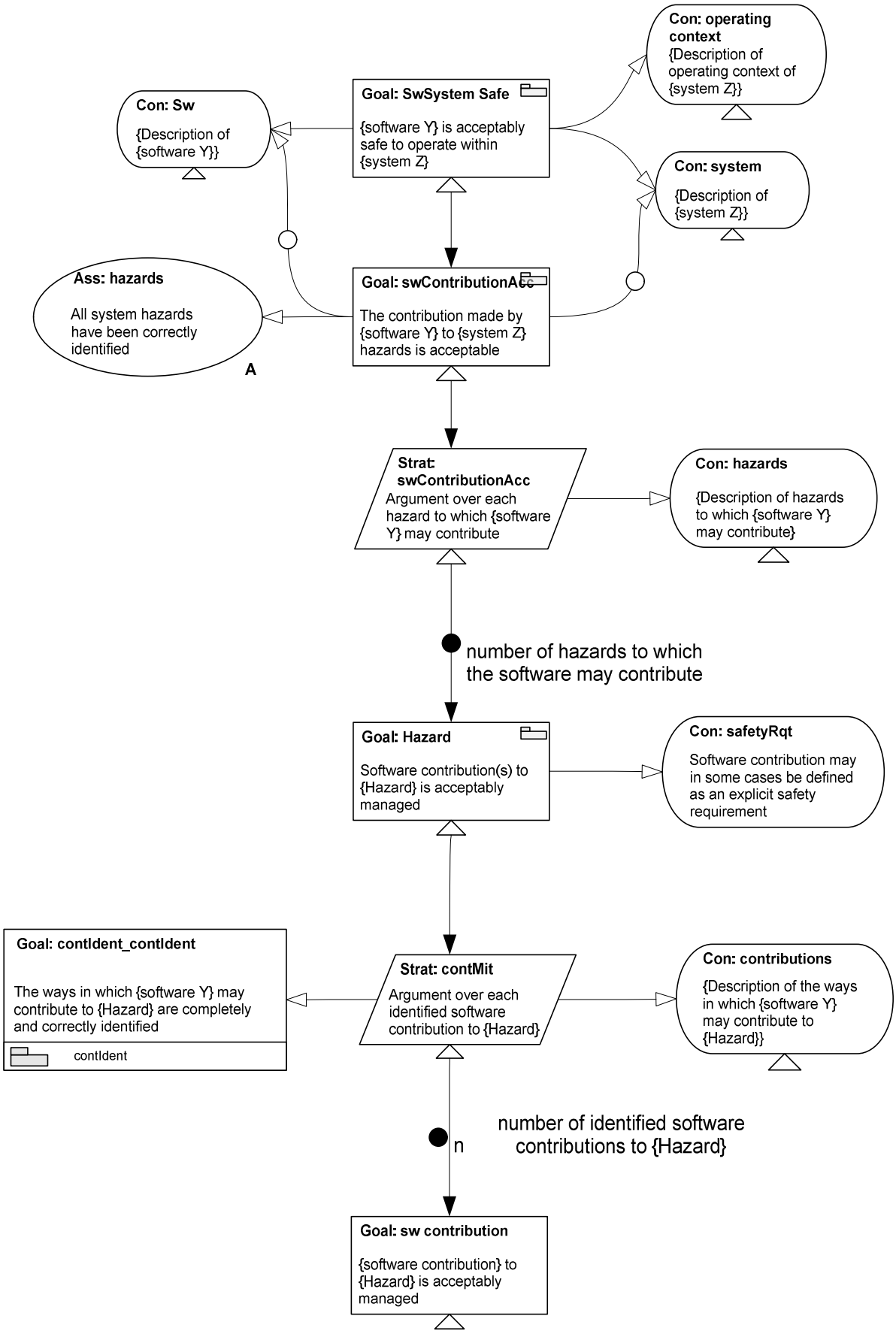

Figure 1 – High Level Software Safety Argument Pattern Structure

PARTICIPANTS

Goal: SwSystem Safe

If a stand-alone software safety argument is being produced then this goal should be used as the top goal in the argument since it clearly sets out the overall objective of the software safety argument. It is necessary to provide the three items of context to make the scope of the software safety argument clear to the reader. This goal has been designated as a public goal to indicate that it may be used as the top goal in the argument.

Goal: swContributionAcc

This goal makes it clear that a hazard directed approach is adopted, by considering the contributions made by the software to the system's hazards. If the pattern is being used as part of a system safety argument, then this goal may provide the link in to that argument (hence a public goal). This would be the case if the system safety argument considers the contribution of the software all in one place. It is not necessary to include the context to provide descriptions of the system and the software if this is already clear from the system safety argument.

Ass: hazards

The system hazards can only be identified at the system level. Identification of system hazards is therefore outside of the scope of the software safety argument. It is acceptable therefore to make this assumption as long as the assumption is demonstrated elsewhere at the system level. If an argument to support this assumption exists with a system safety argument then it would be appropriate to link to that argument at this point instead of making an assumption.

Strat: swContributionAcc

To ensure traceability from the software to system hazards, the strategy adopted is to argue explicitly over each of the hazards identified at the system level.

Goal: Hazard

For each hazard there may be one or more potential contributions from the software identified at the system level. An instance of this goal is created for each of the system hazards to which the software may contribute. At the system level the software will only be considered from a 'black-box' point of view, so the contribution may be identified in the form of high-level functionality, or safety requirements. These contributions would be considered base events at the system level, and would not generally be developed further in a system level argument.

Goal: contIdent

It is necessary to ensure that all the software contributions are correctly identified at the system level. This is crucial to the assurance of the argument as it provides the warrant for the adopted strategy of arguing over the software contributions. This goal provides context to the strategy contMit and must be supported by an argument contained in a separate module (contIdent). Software contributions are often identified as base events in a fault tree analysis performed at the system level. The argument in module contIdent would, in such a case, reason about the rigour and suitability of that analysis.

Goal: sw contribution

An instance of this goal is created for each of the identified software contributions to each of the system hazards. The Software contribution safety argument pattern (section ?) may be used to generate an argument to support this goal.

APPLICABILITY

This pattern should be applied whenever a software safety argument is required as part of a safety case.

CONSEQUENCES

Once this pattern has been instantiated, a number of elements will remain undeveloped and requiring support. Firstly 'Goal: sw contribution' must be supported. The Software contribution safety argument pattern presented in this catalogue can be used to support this goal. In addition, an argument to support 'Goal: contIdent' must also be developed in module contIdent. This argument will be based on analysis performed at the system level, so in some

cases a sufficient argument may exist at the system level which can be used to support this claim.

IMPLEMENTATION

There are a number of different possible top goals for this pattern, as indicated by the public goals. The appropriate top level goal for the argument must be determined through consideration of the structure of any system safety argument which the software safety argument supports. If the pattern is being used to support a system level safety argument, the top goal from this pattern may not actually appear at the top of the overall argument structure. Instead it will appear as a child-goal within the system safety argument. It is important that a stand-alone software safety argument begins with the top goal 'Goal: swSystem Safe' to capture the overall objective of the argument and all the required contextual information.

POSSIBLE PITFALLS

The software contributions may not have been adequately identified at the system level. This may then necessitate further analysis at the system level. It is therefore clearly advantageous to ensure software is considered as part of the system level safety activities.

RELATED PATTERNS

This pattern is supported by the Software contribution safety argument pattern.