3.2. Software Contribution Safety Argument Pattern

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

INTENT

This pattern provides the structure for arguments that the contributions made by software to system hazards are acceptably managed.

MOTIVATION

It is necessary to consider all of the ways in which errors may be introduced into the software which could lead to the software contribution. The software development process used will vary between different projects, however in all cases the software development is undertaken through varying levels of design. At each level the design must satisfy requirements of the higher level. These requirements may be explicitly captured as part of a requirements specification, or identified implicitly from the design itself. In [6] Jaffe et al propose an extensible model of development which captures this relationship between components at different tiers. Figure 1 illustrates the multi-tiered relationship between successively more detailed requirements and design information. Figure 2 illustrates in more detail the relationship among a tier n component's requirements, its design representation, and the tier n+1 requirements of the tier n+1 (sub) components identified in the design representation.

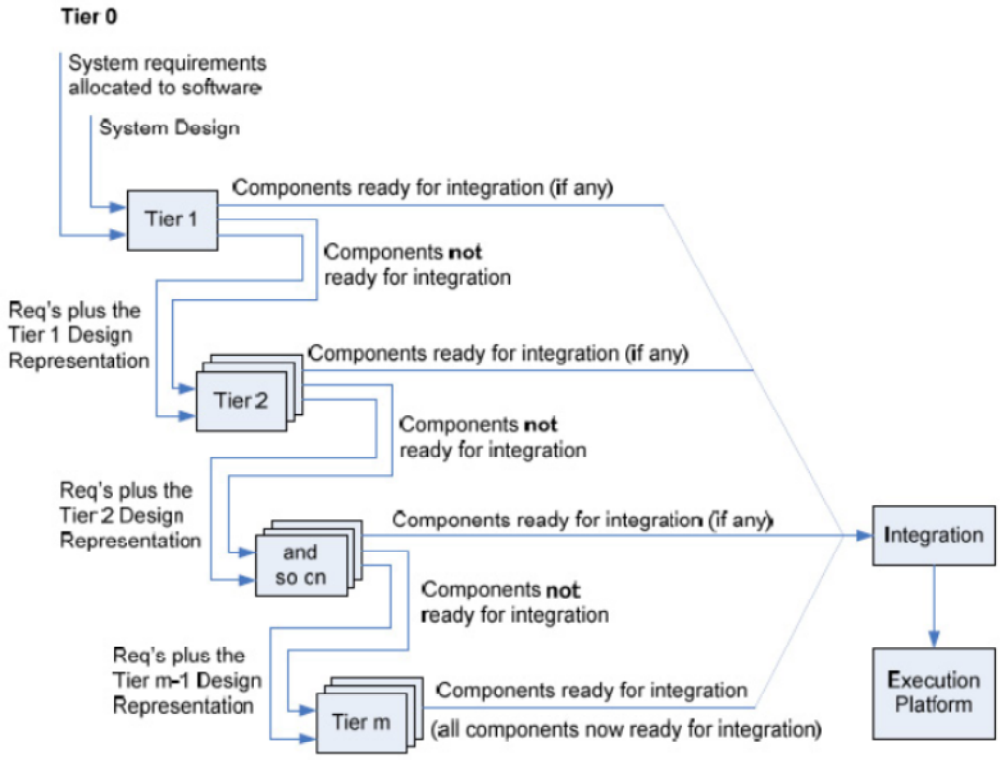

Figure 2 - Illustration of a multi-tiered relationship

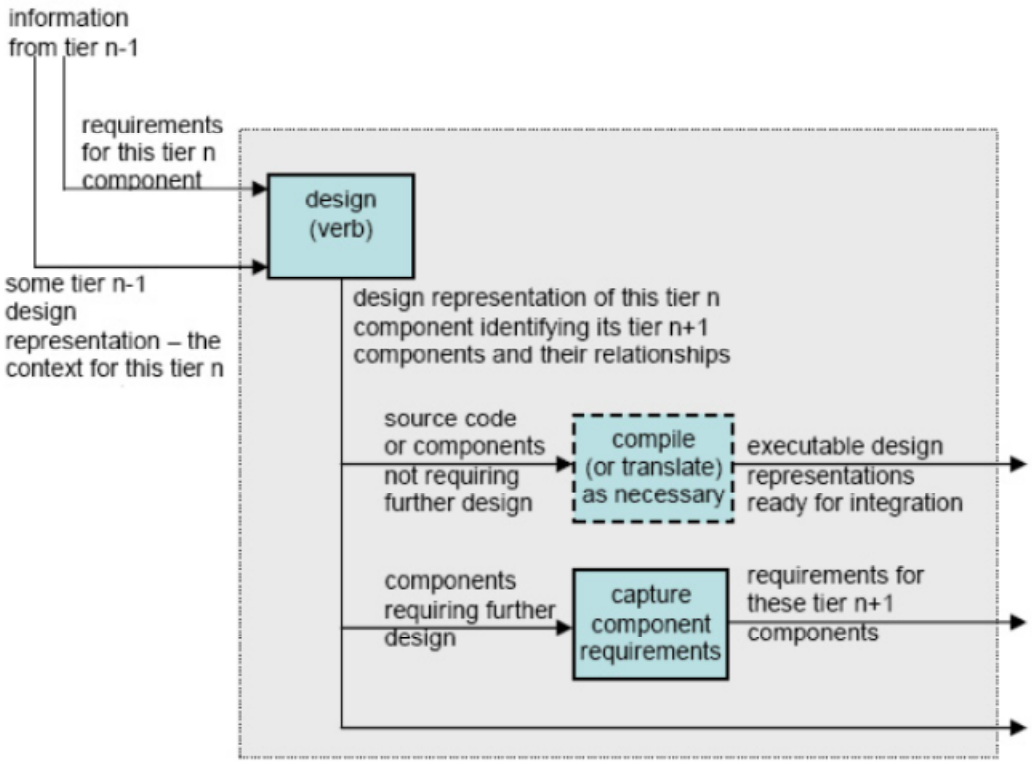

Figure 3 - More detailed illustration for a tier n component

From a safety perspective, it is necessary to ensure that at each tier, the software safety requirements derived at the previous tier are adequately addressed. This involves making design decisions which mitigate potential failures and adequately allocating and decomposing the software safety requirements (SSRs) through consideration of the design at that tier. At each tier it is also possible to introduce errors into the software which could manifest themselves as hazardous failures. It is therefore important in the software safety argument to also consider additional hazardous contributions that may be introduced at each tier.

This pattern therefore reflects the tier model discussed above in order to make it generally applicable to a broad range of development processes.

STRUCTURE

The structure of this argument pattern is shown in Figure 4 below.

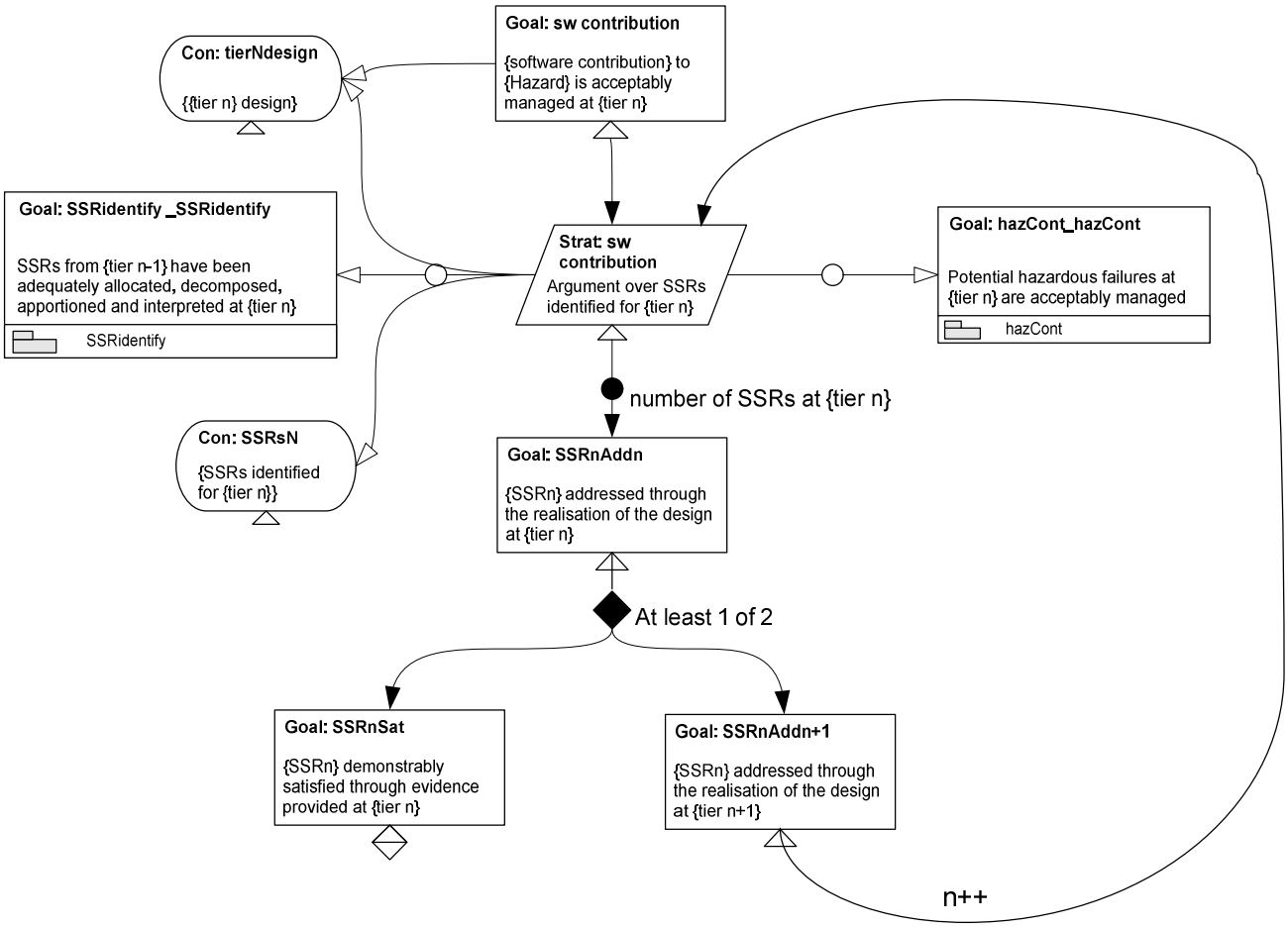

Figure 4 – Software Contribution Safety Argument Pattern Structure

PARTICIPANTS

Goal: sw contribution

An instance of this goal is required for each of the identified software contributions to each of the system hazards. For this top claim in the pattern, {tier n} will in this case refer to the highest tier in the development process. This highest tier is generally referred to as (high-level) software requirements.

Strat: sw contribution

The strategy adopted is to argue over all the SSRs which are identified at this tier. These SSRs are either derived from the DSSRs of the previous tier, or through consideration of additional hazardous contributions that may occur at this tier.

Goal: SSRidentify

The SSRs from the previous tier must be allocated to the tier n design appropriately, having been suitably decomposed where necessary, and correctly apportioned across the design as part of that decomposition. The SSRs may also require interpretation to reflect the tier n design. As part of supporting this goal it is necessary to consider the design decisions that are taken in order to mitigate failures, including mechanisms for failure detection and response. At the highest tier, there are no SSRs from the previous tier, instead, the software contribution itself must be considered. This goal is crucial to the assurance of the argument as it provides the warrant for the adopted strategy of arguing over SSRS identified for tier n. This goal must be supported by an argument contained in a separate module (SSRidentify). The SSR identification software safety argument pattern may be used to generate an argument to support this goal. This goal is optional, since it may not necessarily be required to provide direct traceability at every tier. The decision as to whether this is required at a particular tier must be based on a consideration of assurance. It may be necessary to justify such a decision by providing an argument. The Argument justification software safety argument pattern may be used to provide such an argument. It would be possible, instead of supporting this goal, to simply provide an assumption node stating that SSRs from the previous tier have been adequately allocated, decomposed, apportioned and interpreted. This would however significantly

reduce the assurance achieved, so the impact of such a decision must be considered (see section 3.2).

Goal: hazCont

At any tier in the development there is the possibility of introducing additional contributions to hazards due to errors made at that tier. This goal claims that such potential hazardous contributions are addressed through the specification of additional SSRs. Supporting this goal requires that the potential hazardous contributions at tier n are adequately identified, and that SSRs sufficient to address those hazardous contributions are specified. This goal is crucial to the assurance of the argument as it provides the warrant for the adopted strategy of arguing over SSRs identified for tier n. This goal must be supported by an argument contained in a separate module (hazCont). The Hazardous contribution software safety argument pattern may be used to generate an argument to support this goal. This goal is optional, since it may not necessarily be required to identify hazardous contributions at every tier. The decision as to whether this is required at a particular tier must be based on a consideration of the impact on assurance. It may be necessary to justify such a decision by providing an argument. The Argument justification software safety argument pattern may be used to provide such an argument. It would be possible, instead of supporting this goal, to simply provide an assumption node stating that DSSRs have been correctly identified at tier n to address the identified potential additional hazardous contribution. This would however significantly reduce the assurance achieved, so the impact of such a decision must be considered (see section 3.2).

Goal: SSRnAddn

An instance of this goal is created for each SSR identified at tier n (represented as SSRn). There is an option for how this goal is supported. It can be supported by either, or both of goals 'SSRnSat' and 'SSRnAddn+1'. It may be necessary to justify such a decision by providing an argument. The Argument justification software safety argument pattern may be used to provide such an argument.

Goal: SSRnAddn+1

It is possible to demonstrate that the SSRs at tier n are addressed by showing traceability down to the subsequent tier of development. The argument then

continues through a further instantiation of 'Strat: sw contribution'. {tier n+1} then becomes {tier n}.

Goal: SSRnSat

It is possible at any tier to provide verification evidence of the satisfaction of the SSRs for that tier. This may be, for example, testing or analysis performed at that tier. Not all software is subject to the same number of tiers of development. Also, not all aspects of any particular software are necessarily developed over the same number of tiers. It is therefore also possible for implementation to occur at any tier. At the tier of implementation it is possible to provide argument and evidence to demonstrate that the SSR is satisfied by the implementation. As discussed in 4.3.6, detailed guidance on the development of the argument to support this goal will be the subject of future work.

APPLICABILITY

This pattern should be applied as part of any hazard-directed software safety argument.

CONSEQUENCES

Once this pattern has been instantiated, a number of elements will remain undeveloped and requiring support. 'Goal: SSRIdentify' must be supported. The DSSR identification software safety argument pattern presented in this catalogue can be used to support this goal. 'Goal: hazCont' must be supported. The Hazardous contribution software safety argument pattern presented in this catalogue can be used to support this goal. Finally 'Goal: SSRnSat' must be supported. As discussed, detailed guidance on the development of the argument to support this goal will be the subject of future work.

IMPLEMENTATION

This pattern should be instantiated as part of a software safety argument. An instantiation of 'Goal: SSRIdentify' must be created for each identified software contribution to each system hazard. {tier n}, and {tier n+1} must be instantiated with the names of the relevant tier. Note that as the argument is developed over multiple tiers, {tier n} will refer to different tiers. {SSRn} is used to refer to a SSR at tier n, and should be instantiated with the SSR itself or a

unique identifier for the SSR. Note that in this pattern the looping link represents a repeating pattern of argument, and would not appear in such a manner in an instantiated argument.

POSSIBLE PITFALLS

Whilst acknowledging that in many cases not all the optional goals may be provided at each tier, it is also important to note the significance of this pattern on the achieved assurance. Assurance deficits introduced in instantiating this pattern can have a potentially large impact. In such cases the additional support may prove necessary. It is therefore important that the assurance impact of decisions taken at each tier of development are fully considered, to avoid additional work at a later date. (See section 3.2).

RELATED PATTERNS

Consideration should be given to the application of the Argument justification software safety argument pattern wherever significant decisions about how to instantiate the optional aspects of this pattern are made. The Argument justification software safety argument pattern should be instantiated in context to this pattern to justify the acceptability of any residual assurance deficits as a result of the instantiation decisions. This pattern supports the High-level software safety argument pattern. Support for 'Goal: SSRIdentify' and 'Goal: hazCont' can be provided using the SSR identification software safety argument pattern and the Hazardous contribution software safety argument pattern respectively.