

Assurance, Confidence and Software Safety

Dr. Richard Hawkins

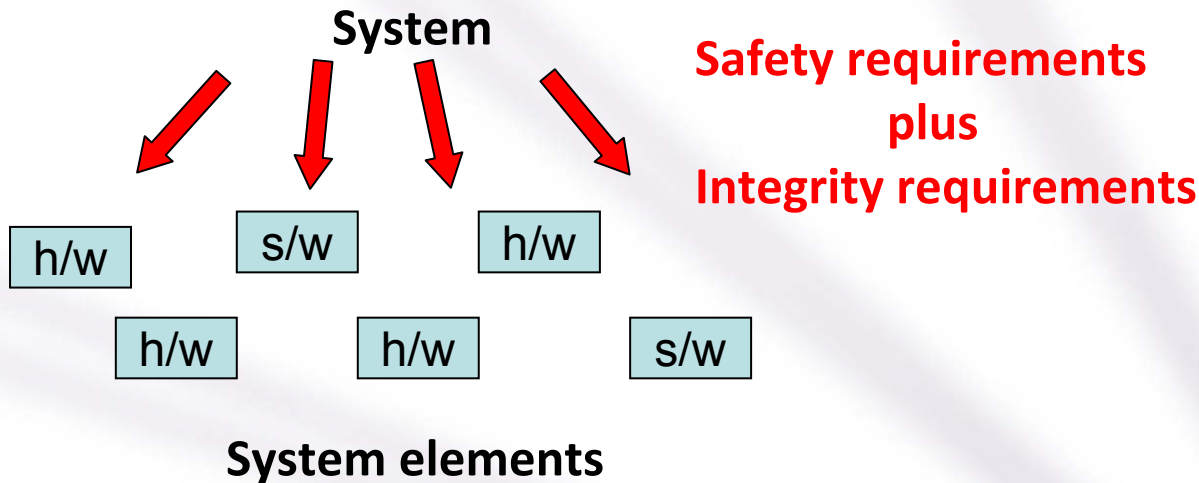


5th May 2009

Background to the problem



← Safety/hazard analysis



- For s/w very difficult to demonstrate safety requirements to integrity $> \sim 10^{-3}$
- For s/w need different approach, since can't *directly* demonstrate an integrity

Software Safety Arguments

- Traditionally safety of software aspects of systems demonstrated using a prescriptive approach
 - Process defined in a standard
 - Process varied according to risk or criticality of sw failure
- Move towards a goal-based approach
 - RA does not prescribe a method
 - Responsibility of developer to demonstrate safety to RA
 - Production of a software safety argument
- Why move to a goal-based approach?
 - Does controlling process necessarily control hazardous contribution?
 - Perhaps...but implicit
 - The developer will always know what is best to do to demonstrate the system is safe
 - Increased flexibility allows easier adoption of new techniques and technologies

The challenge

- But, creating compelling software safety arguments is difficult and costly
- Interviewed a number of stakeholders to find out the key challenges
 - Difficult to determine what activities it was necessary to undertake to support the software safety case
 - Particularly, how to ensure what you do is appropriate for the level of risk.
 - When have you done enough?
 - How can sufficiency of the safety argument be judged?
 - How can you determine up front what you will need to do?
 - So that you can manage the activities

Addressing risk

- It is necessary that the safety argument produced is commensurate with the system risk
- Goal-based approach explicitly addresses risk reduction
- But we still need to determine if mitigations put in place are sufficient for given risk
- There have been some attempts to define the evidence required for different risks
 - However relationship between generated evidence and risk reduction achieved is unclear
 - So justifying sufficiency remains very difficult

Assurance

- Why is it so difficult?
- Safety arguments are very rarely deductive
 - If we know premises to be true, then we will also believe the conclusion with certainty
- Mostly we have inductive arguments
 - We can't demonstrate the conclusion of the argument with certainty, only with probability
- This probability represents the level of confidence we have in truth of the claim
- The term *assurance* is often used in safety arguments
 - *The assurance of a claim is justifiable confidence in the truth of that claim*

Assurance

- Lots of factors affect assurance..
 - Assumptions made
 - Inductive gap
 - How strongly do sub-claims or evidence give reason to believe the claim is true?
 - Trustworthiness of evidence
 - What is the quality of the evidence?
 - How well does it meet its intent?
 - Visibility of system and environment information
 - etc
- These subjectivities are always there
 - Present in a prescriptive approach, but left implicit
- By reasoning explicitly about them, it is easier to justify

Compelling Software Safety Arguments

- Necessary to demonstrate that sufficient assurance has been achieved
- Must consider assurance throughout the development of the software safety argument
- Guidance on this split into two parts
 - Software safety argument pattern catalogue
 - Assurance based argument development method

Software Safety Argument Pattern Catalogue

- Capture good practice for compelling software safety arguments
 - Based upon
 - Existing patterns
 - Current practice for software safety arguments
- Developed to provide flexibility
 - Instantiable for a wide range of systems
 - Diverse development processes
 - Different hazards and safety requirements
 - Etc..
- To be sufficiently compelling the correct implementation decisions must be made
 - Patterns must be instantiated within the framework of the assurance based argument development method (more later...)

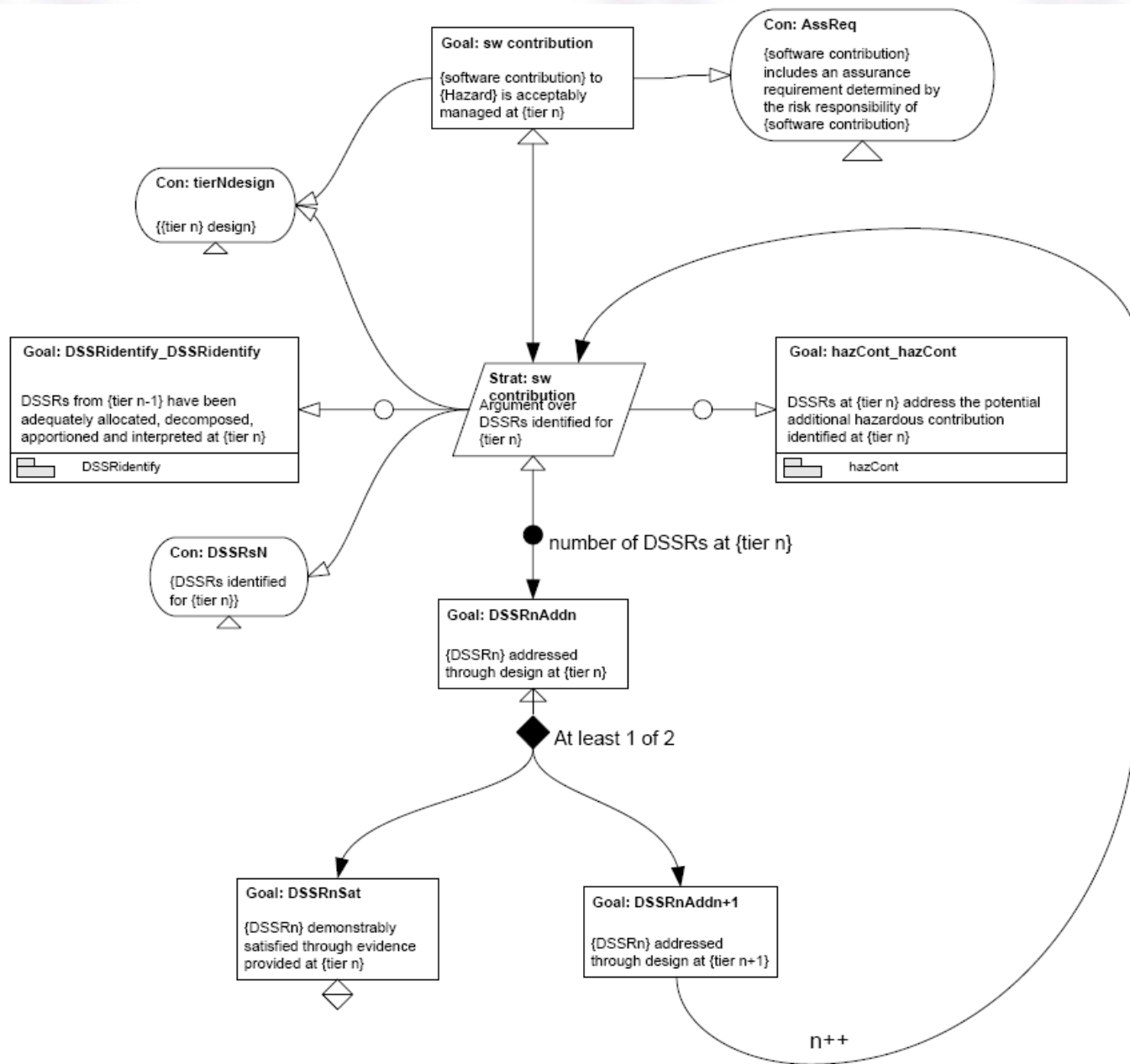
Software Safety Argument Pattern Catalogue

- Patterns currently provided in the pattern catalogue...
- High-level software safety argument pattern
 - High-level structure for a software safety argument
- Software contribution safety argument pattern
 - Arguments that the contributions made by software to system hazards are acceptably managed
- DSSR identification software safety argument pattern
 - Arguments that DSSRs from one 'tier' are captured at the next
- Hazardous contribution software safety argument pattern
 - Considers additional hazardous contributions at each tier
- Strategy justification software safety argument pattern
 - Argument that the adopted strategy is acceptable

Software Contribution Safety Argument Pattern

- Must consider all ways in which errors introduced into software could lead to the software contribution
- Different development process used on different projects
 - Always have various 'tiers' of design
- At each tier must address requirements of the higher level
 - DSSRs from the previous tier must be adequately addressed
 - Consider additional hazardous contributions that may be introduced at each tier
- Instantiation decisions made here will have large impact on assurance

Software Contribution Safety Argument Pattern

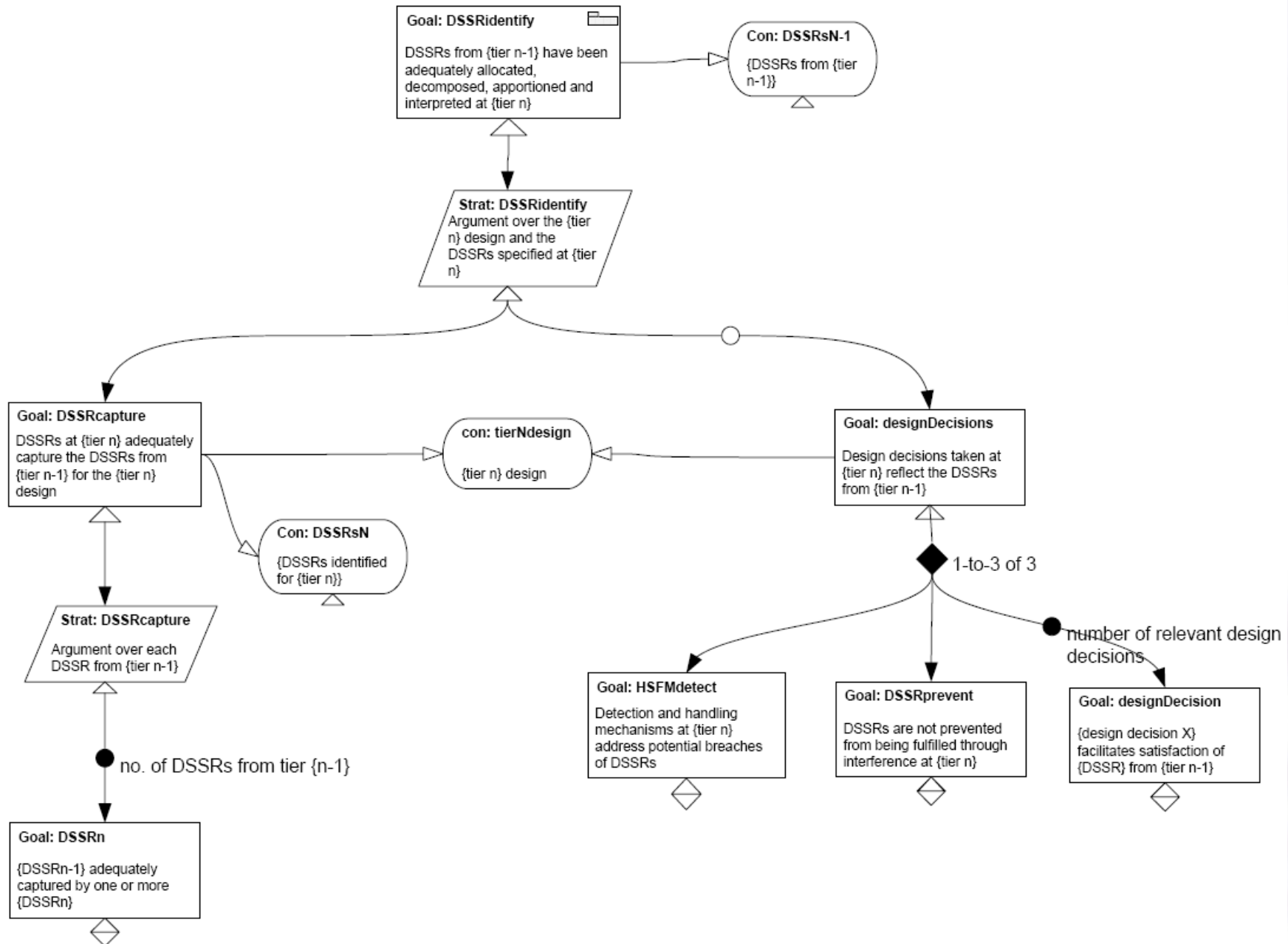


DSSR Identification Sw Safety Argument Pattern

- (DSSRs) from a previous tier of development adequately captured at the next tier of development
 - Design mitigations
 - Allocate and decompose DSSRs
 - Define additional DSSRs

- Don't *necessarily* need to instantiate for every tier but..
 - Violates traceability requirements
 - Increases uncertainty
 - Must be able to justify this is acceptable

DSSR Identification Sw Safety Argument Pattern



Hazardous Contribution Sw Safety Argument Pattern

- Potentially hazardous failures could be introduced at each tier
 - Must identify HSFM at that tier
 - FHA
 - HAZOP
 - Etc...
 - Must address each identified HSFM
 - Definition of further DSSRs
- Don't *necessarily* need to instantiate for every tier but...
 - Violates traceability requirements
 - Increases uncertainty
 - Must be able to justify this is acceptable

Strategy Justification Sw Safety Argument Pattern

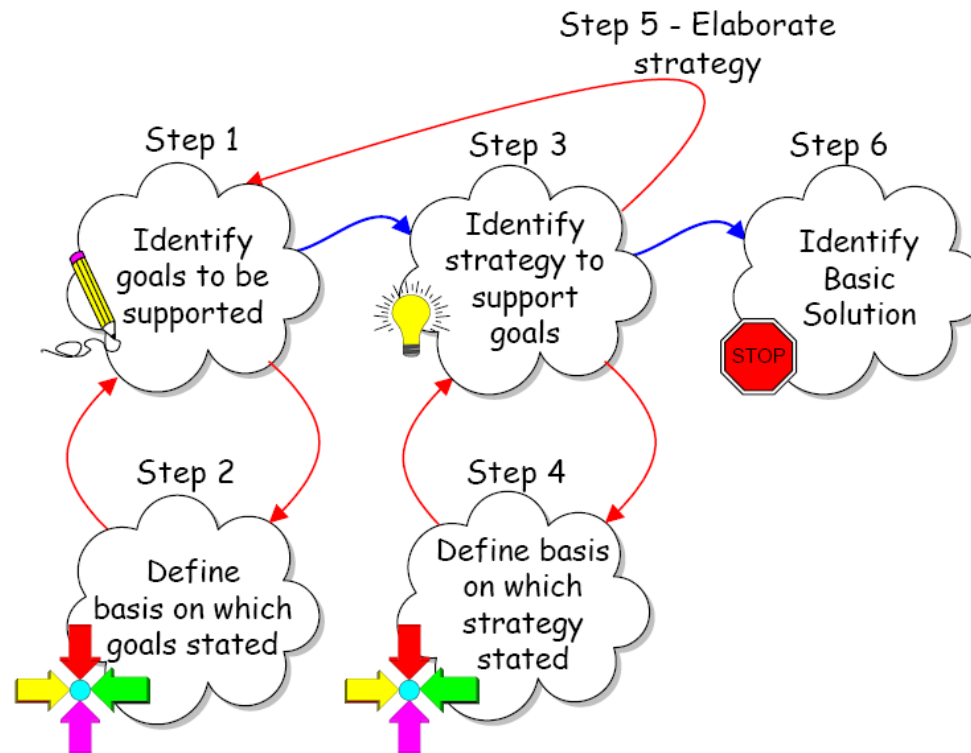
- The strategy adopted is acceptable from assurance point of view
 - Justify implementation decisions made are appropriate
- The confidence achieved in the claim is acceptable
 - Provides explicit justification
 - Based on ACARP assessment
- Can be used to justify any strategy for which justification may be required to convince a reader
- Pattern is used in context to the strategy to which it relates
- Will look at this in detail after ACARP discussion...

Assurance Based Argument Development Method

- Even if using patterns for guidance how can we be sure the argument is sufficiently compelling?
- Must explicitly consider assurance throughout argument development
- At every step in constructing the argument it is inevitable that information will be lost
 - Defining the safety claims
 - Deciding on strategy (argument approach)
 - Identifying assumptions and context
 - Providing evidence
- Losing information increases uncertainty, which affects assurance
 - Assurance deficits
- To construct compelling arguments must understand where assurance deficits come from

Sw safety argument development method

- There is an existing safety argument development method



- This can be used to develop software safety arguments
 - Assurance is not explicitly considered
 - Potential for assurance deficits

Extended 6-step method

- To extend the 6-step method
 - Considered how assurance deficits may occur at each step
 - Use this to inform decisions about how to construct the argument

- Perform deviation analysis on each of the steps
 - No or None - More - Less - As well as - Part of - Other than – Reverse

- Apply and interpret guidewords for each step

- Consider deviation effect on assurance
 - What information is being lost?
 - How would that information affect assurance?
 - Is it worth knowing that information?

Consideration of Assurance During Argument Construction

Step	Purpose	Assurance impact			
1. Identify goals to be supported	To clearly and unambiguously state the goals to be supported.	More - If in stating the goal, an attempt is made to claim more than it is actually possible to support with the available evidence, then the assurance that can be achieved in that goal will inevitably be low.	Less - The stated goal may claim less than is actually required to support the argument. Although in this case it may be easier to achieve higher confidence in the stated goal, this confidence will not result in the expected assurance in the parent goal, since the claim is insufficient to support the conclusion.	As Well As - A strategy or solution may be erroneously included in the claim. This can inadvertently constrain potential options for addressing assurance deficits.	Other Than - The claim made may not actually be that in which assurance is required. Assurance may be lost through failing to correctly capture the true intent of the claim.
2. Define basis on which goals are stated	To clarify the scope of the claim, to provide definitions of terms used, to interpret the meaning of concepts.	None - Any claim is only true or false over a particular scope. If the scope of the claim is unclear, due to lack of context, then the level of truth or falsity of the claim becomes more difficult to determine. This increases the uncertainty associated with the assurance in that claim, and therefore makes it more difficult to determine the assurance.	More - The scope of the claim as defined by the context may be too narrow. The result of this is that although a certain level of assurance may be achieved over the scope defined by the context, the narrowness of the scope limits that in which confidence is achieved.	Less - The scope of the claim is too loosely defined. The effect of this would be similar to having no context at all, in that it leads to uncertainty, and a corresponding reduction in assurance.	
3. Identify strategy to support goals	To identify how a goal can be adequately supported.	<p>More, Less, Other Than - This step of the safety argument process is the most crucial for the assurance achieved since it is at this step that the decisions are made about which strategy should be adopted to support each claim. Assurance is lost at this step if the proposed strategy does not provide sufficient support to the goal. This could happen for two reasons.</p> <ul style="list-style-type: none"> • Firstly the inductive gap may be too large. If this is the case, then even if the premises are believed, it doesn't provide sufficient confidence in the truth of the conclusion. • Secondly the fundamental beliefs upon which the strategy is based may be open to question. In such a case the premises may not provide confidence in the conclusion. 			
4. Define basis on which strategy is stated	To identify any assumptions upon which the sufficiency of the strategy depends.	No, Less - It is inevitable that some assumptions will be made during the development of any safety argument, however these assumptions may not always be explicitly captured. Any assumptions that are left implicit introduce uncertainty, and reduce	More - All assumptions are, by definition, unsupported. The argument holds only on the basis that the assumptions are true. If there is a lack of confidence in the truth of the assumptions, then this will also result in a lack of confidence in the truth of the claim. It is	Other Than - Assumptions may be stated which are not actually true. Any false assumptions undermine the whole basis upon which the argument is made.	

Consideration of Assurance During Argument Construction

		assurance.	therefore recommended, for any assumptions that may be open to any significant doubt, that an argument is presented, rather than an assumption.		
	To provide justification for why a particular strategy is being used.	No, Less - No justification is provided as to why the adopted strategy is sufficient. This can result in a loss of assurance, since there may be a lack of confidence in the sufficiency of that strategy. It is important, if it's likely that the justification may be unclear, not to leave it implicit, but to explicitly record the justification in the argument. The ACARP approach may be used to provide such a justification.	More - Although not leading to a loss of assurance, it is important to note that providing an argument to justify the strategy chosen in each decomposition in the argument is not necessary. For many strategies, the justification will be obvious to the reader and may be left implicit.		
5. Elaborate strategy	Specify the goals that implement the chosen strategy.	Less, As Well As, Part Of - The strategy that is actually implemented does not fully and accurately reflect the one that was chosen. Assurance may be lost at this step, even though a chosen strategy may be considered acceptable.			
6. Identify basic solution	Identify the solutions which provide adequate support to the goal.	Less - The solution provides less confidence in the goal being supported than is required. Assurance is lost at this step if it is unclear why the evidence gives confidence in the goal being supported. It may be unclear because: <ul style="list-style-type: none"> • there may be an inductive gap between the claim and the evidence (the nature of the evidence does not provide a compelling reason to believe the claim is true) • there is uncertainty about the trustworthiness of the evidence itself. Note that evidence which is untrustworthy will undermine assurance even in the situation where there is a deductive relationship between the claim and the evidence. 	Other than - Counter evidence is any evidence which undermines the confidence in the claim being made. The presence of counterevidence does not necessarily mean that the argument is inadequate. It simply means that the confidence in the claim may now be lower than it was before the counter evidence was identified. It is necessary to determine the impact of the counter evidence on the claim's assurance. In many cases it may still be possible to make a sufficiently compelling argument despite the identification of counter evidence, particularly where there are mitigations which limit the uncertainty caused by the counter evidence. If counter-evidence isn't correctly identified then the potential effect on the assurance of the argument cannot be determined. It is important to justify that there is sufficient confidence that relevant counter evidence has been identified.		

ACARP

- Possible to increase assurance in a claim by gaining more relevant information - address assurance deficit
 - But is it cost-effective to do so?
 - Diminishing returns?
 - How do we know when we've increased confidence sufficiently?
- DS 00-56 Issue 4 Part 2 Annex B states...
 - *“B1.1 – The goal of risk management as defined by this standard is to show that safety risks can be tolerated and are at levels that are ALARP”*
 - *“B3.2 – [For systems containing complex electronic elements]...much of the effort only improves confidence that requirements have been met. In applying ALARP, the confidence achieved should be proportionate to the risk.”*
- This leads us onto a consideration of ACARP
(As Confident As Reasonably Practicable)

Assurance Deficits

- Developer must be able to justify they are ACARP in the truth of the claim
- Ensuring sufficient confidence is achieved requires that all assurance deficits are acceptably managed
- Potential assurance deficits may be identified from
 - Assurance based argument development method
 - The patterns
- For all identified assurance deficits
 - Consider if they're acceptable
 - Attempt to address the deficit
 - Justify any residual assurance deficit

Impact Assessment

- To determine if identified assurance deficit should be addressed
 - Must consider the *impact* of assurance deficit
- What is effect of not having the information on the claim being supported?
 - What is still assured and what isn't?
- How bad would it be if the claim was undermined in this way?
- Important to consider in terms of risk
 - Only through considering risk can be know how bad it is
 - Importance to overall system safety

ACARP Assessment

■ Can use ACARP to categorise impact of assurance deficits

○ *Intolerable*

- Potential impact on the claim of assurance deficit cannot be justified under any circumstances

○ *ACARP*

- Assurance deficit is tolerable only if the cost of taking measures to address assurance deficit is grossly disproportionate to the benefit
- The greater the impact of the assurance deficit, the more, proportionately, system developers are expected to spend to address it

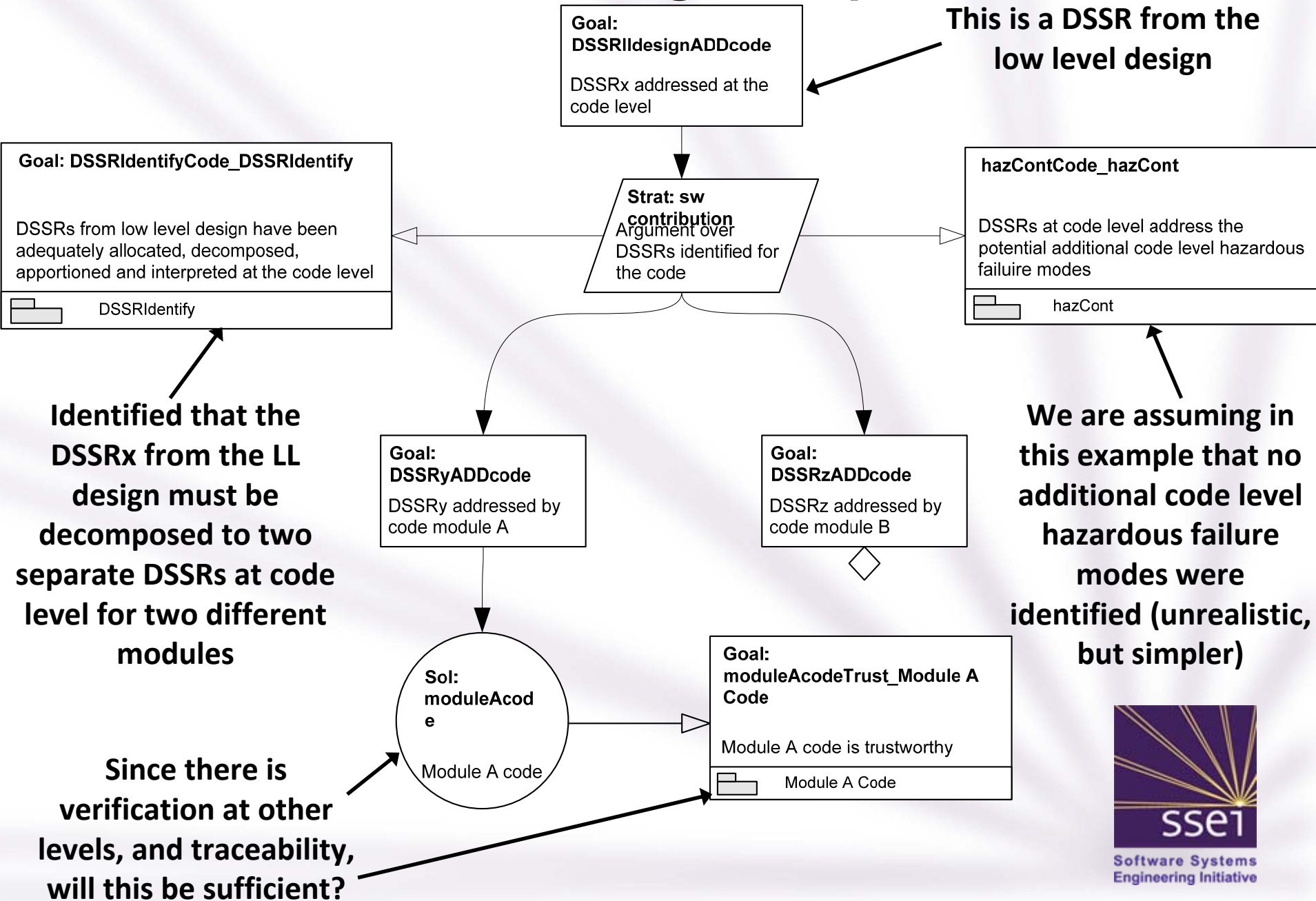
○ *Broadly acceptable*

- Impact of assurance deficit is negligible, further increases in confidence need not be sought

Justifying Sufficiency

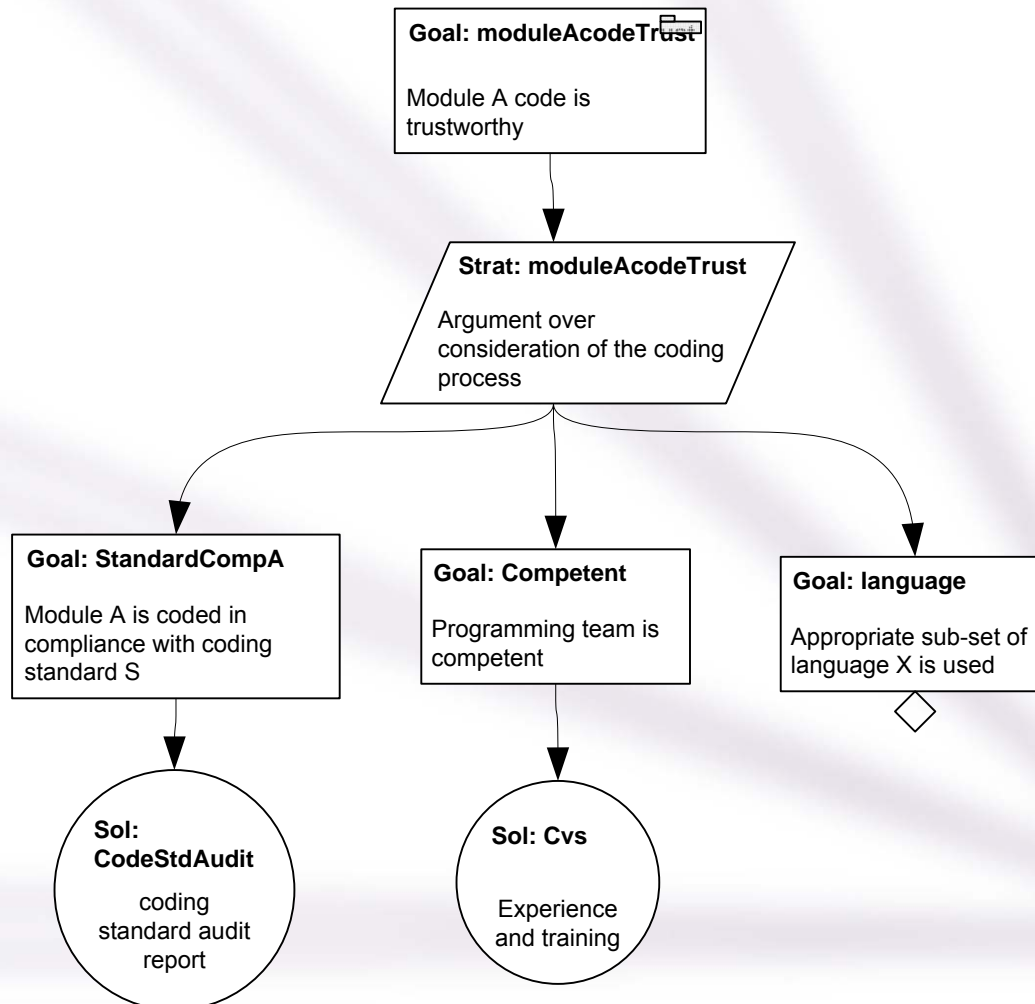
- Addressing assurance deficit requires ‘buying’ more information relevant to the safety claim
 - Is it worth spending the money to get that information?
- Demonstrating ACARP requires that both the cost and impact of addressing assurance deficits be determined
 - To judge if the cost is grossly disproportionate to the impact
- In theory could do formal cost-benefit analysis based on quantitative assessment of
 - Cost of available options
 - Costs associated with potential impact
- In most cases for ACARP, qualitative approach is more appropriate
 - But relies on providing explicit justification why residual assurance deficits are acceptable
 - Justification based on (qualitative) ACARP assessment
 - Where appropriate provide an argument

Unit Testing Example



Unit Testing Example

- We do have some confidence in the truth of the claim ‘DSSRy addressed by code module A’
 - Provided by the trustworthiness argument...



Unit Testing Example

- This is providing some confidence that the code of module A is free from errors
- To consider if the confidence is sufficient, must consider what additional information could be provided relevant to the claim
 - What is the potential assurance deficit here?
- We could provide information about any errors that were made in implementing the module...
- Unit testing could provide information about this
- What is the impact of this information?
 - What is the effect of not doing unit testing on Goal:DSSRyADDCode?
 - How bad would it be if there was no way of knowing about errors introduced during implementation?

Unit Testing Example

- Without unit testing there is...
 - No mechanism for identifying any errors which are introduced at the code level
 - No way of determining whether the errors could affect the achievement of DSSRy
- The effect on risk can be determined by considering the potential hazardous effect of unidentified errors
- The potential hazardous effect is that the safety requirement (DSSRy, DSSRx, and upwards) is not met in operation
- Impact reflected by risk at system level
 - Defines relative importance of affected safety requirements to system safety
- Impact will also depend upon relative assurance of module A code free from errors

Unit Testing Example

- Not performing unit testing contributes to the DSSR not being met...
 - iff there is an error which unit testing would've identified which leads to the DSSR failure
- The more confidence there aren't errors in code module
 - The less likely it is that there will be an error which leads to DSSR failure
 - The impact of not doing unit testing is reduced
- Assurance in this other aspect of the support for Goal:DSSRyADDCode can reduce the impact of the assurance deficit

Unit Testing Example

- If DSSRy has low importance to system safety (low risk)
- And have high assurance that code is error free
 - based on argument of trustworthiness of code
- Potentially could justify that...
 - High cost of unit testing is grossly disproportionate to benefit gained
 - Since impact of addressing assurance deficit is low
- Where impact determined to be higher such justification may not be possible
 - Unit testing would be considered reasonably practicable
- Other strategies could increase confidence further through providing further information relating to
 - Presence of errors in the code
 - Lack of errors in the code (trustworthiness)

Unit Testing Example

- Other methods could provide similar information to unit testing
 - E.g. static code analysis
- Important to consider what *additional* information is provided relative to the claim
- Static analysis will only increase confidence further if providing information unit testing does not
- Must consider
 - Weaknesses or limitations of unit testing
 - The nature of the claim
 - Different DSSRs may require different support
 - E.g. timing vs omission
 - Some may require a combination of techniques to provide required information

Unit Testing Example

- Could also gain additional assurance in the trustworthiness of the code
 - Provide more information about the rigour of the processes used
- Provides the opportunity to perform trade-offs between
 - Cost of increasing confidence in lack of errors
vs
 - Cost of increasing confidence in identification of errors
- Where will most benefit be gained?
- Where impact of assurance deficit is high, such trade-offs are unlikely to be justifiable however

Unit Testing Example

- The impact of providing unit testing depends upon the effect of the information it provides in support of the claim
- But also depends on its trustworthiness
 - Are the test team independent of the development team?
 - Were the processes for generating, executing and analysing test cases
 - Systematic and thorough
 - Implemented with rigour
 - Etc..
- Possible to provide a trustworthiness argument for unit testing as well

Software Systems Engineering Initiative

www.ssei.org.uk



5th May 2009