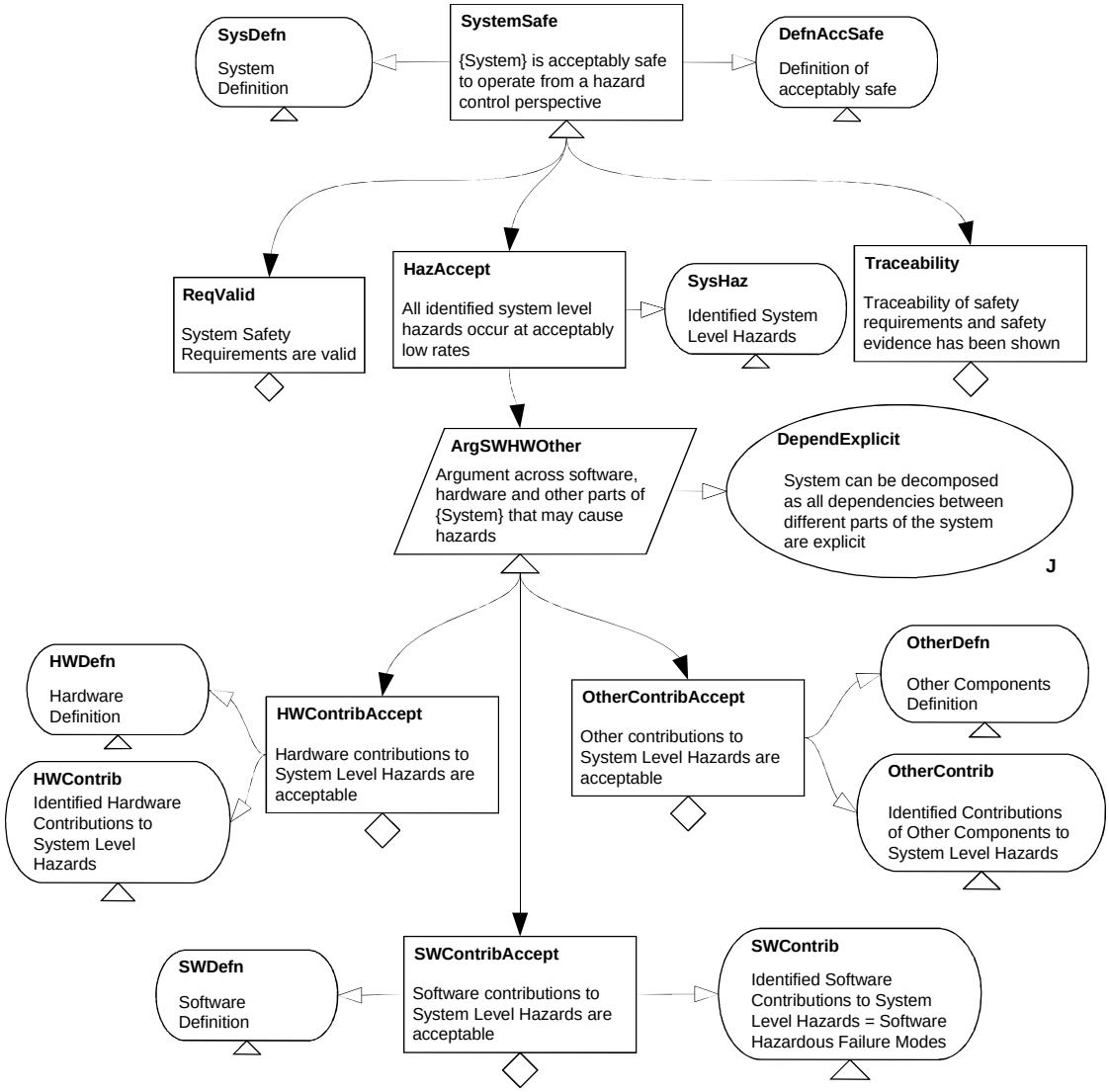
A.3 Catalogue Patterns

Component Contributions to System Hazards			
Author(s)	Rob Weaver, John McDermid, Tim Kelly		
Created	18/09/00	**Last Modified**	20/04/04

Intent	The intent of this pattern is to provide a top level decomposition for the safety argument of a system. In particular, the pattern provides the context for a software safety argument constructed from the Software Safety Pattern Catalogue. The focus for the argument is the identification of hazards and the assessment of the associated risks.
Also Known As	
Motivation	This pattern identifies the three main claims which must be satisfied to show system safety; Valid Safety Requirements, Acceptable Levels of Risks, and Traceability of Safety Requirements and Safety Evidence. The pattern provides a suitable context and approach for developing a software safety argument.

Structure



Participants

SystemSafe	The overall objective of the argument – to provide sufficient support for the claim that the System is acceptably safe to operate.
SysDefn	This model should give a clear definition of the system. From the model it should be possible to identify the system level hazards.
DefnAccSafe	To be able to argue that the claim is upheld, it is necessary to give a definition for the term 'acceptably safe'. This may come from a standard or regulatory body. The definition will be the initial basis from which hazard assessment is made and an argument is generated with respect to the acceptability of the hazards**.**

	ReqValid	This claim asserts that the identified set of safety requirements is applicable (in the correct context) to the system, that they are complete and they are not mutually exclusive.
	HazAccept	This claim asserts the goal that all hazards at the system level have a risk which is acceptably safe as defined by **DefnAccSafe.**
	SysHaz	This context identifies the System Level Hazards upon which the **HazAccept** claim is based. These hazards form a hazard log, which identifies all unsafe behaviours of the system within its operating context.
	Traceability	This claim asserts that it is explicitly visible that the safety requirements have been satisfied through the safety evidence. This enables verification of the complete implementation of the system.
	ArgSWHWOther	This argument decomposes the System Level Hazards across the Hardware, Software and Other Parts of the system. This identifies what part(s) of the system contributes to each individual hazard.
	DependExplicit	The argument **ArgSWHWOther** is justified, so long as the dependencies between Hardware, Software and Other Parts of the System are explicitly documented. This encapsulates the mitigation of particular component failures through other means.
	HWContribAccept	This claim asserts that the Hazards associated with the Hardware component of the system are safe with respect to the definition given in **DefnAccSafe**.
	HWDefn	This Hardware Definition should give a clear description of the system hardware. From the model it should be possible to identify the hardware contributions to system level hazards.
	HWContrib	This context gives the safety requirements which are related to the hardware. These can be either through hardware causes or through derived requirements due to cross dependencies.

	SWContribAccept	This claim asserts that the Hazardous Functions associated with the Software component of the system are safe with respect to the definition given in **DefnAccSafe**.
	SWDefn	This Software Definition should give a clear description of the system software. From the model it should be possible to identify the software contribution to system level hazards.
	SWContrib	This context gives the safety requirements which are related to the software. These can be either through software causes or through derived requirements due to cross dependencies.
	OtherContribAccept	This claim asserts that the Hazards associated with the Other components of the system are safe with respect to the definition given in **DefnAccSafe**.
	OtherDefn	This Other Components Definition should give a clear description of the other components of the system. From the model it should be possible to identify the contribution of other components to system level hazards.
	OtherContrib	This context gives the safety requirements which are related to the other components of the system. These can be either through other component causes or through derived requirements due to cross dependencies.
Collaborations	• The **SysDefn** model should be suitable for identifying the System Level Hazards for **SysHaz.** • The **HWDefn**, in combination with **SysHaz,** should be suitable for identifying the hardware contributions to system level hazards for **HWContrib**. • The **SWDefn**, in combination with **SysHaz,** should be suitable for identifying the software contributions to system level hazards for **SWContrib**. • The **OtherDefn**, in combination with **SysHaz,** should be suitable for identifying the other components contributions to system level hazards for **OtherContrib**. • **HazAccept**, **HWContribAccept**, **SWContribAccept**,	

	OthContribAccept and **SWHazAccept** are all dependent on the definition of acceptably safe in **DefnAccSafe.** • **HWContrib, SWContrib** and **OtherContrib** discharge the justification given in **DependExplicit**.
Applicability	The starting point of this pattern is to have clearly identified the components of the overall system, and their functional contributions to the overall system are understood. In order to apply this pattern it is necessary to have access to a definition of 'acceptably safe' for the **DefnAccSafe** context. This definition is typically provided by the appropriate regulatory authority, standards or through investigations by safety engineers, including discussions with customers. This definition should encapsulate some form of ALARP consideration, which would permeate through the rest of the pattern and argument. System-level and Component-level (Software, Hardware and Other) hazard analysis are required to determine the contributions of the components to system hazards.
Consequences	After instantiating this pattern a number of undeveloped goals will remain: • **ReqValid** & **Traceability** In accordance with the main objective of the pattern, these goals must be developed to give a complete safety argument for the system. • **HWContribAccept**, **SWContribAccept** & **OthContribAccept** To complete the decomposition of **ArgSWHWOther** these three goals need to be decomposed and satisfied. As this pattern provides context for the development of a software safety argument, a pattern for the satisfaction of **SWContribAccept** is identified in the Related Patterns Section
Implementation	This pattern should be instantiated in a Top Down fashion. All goals, contexts and models should be instantiated before continuing to a lower level in the pattern. **Possible Pitfalls** • Not identifying all possible system level hazards may lead to

	missing software safety requirements, which in turn may lead to software failure modes being missed. • Not identifying all dependencies between software, hardware and other parts of the system may cause derived safety requirements to be missed. This would lead to assumptions about mitigation not being discharged.
Examples	None provided at this stage.
Known Uses	
Related Patterns	*Hazardous Software Failure Mode Decomposition* – This pattern can be used to decompose the undeveloped goal **SWContribAccept**. This pattern forms part of a software safety argument pattern catalogue, which includes the following patterns: *Component Contributions to System Hazards* *Hazardous Software Failure Mode Decomposition* *Hazardous Software Failure Mode Classification* *Software Safety Argument Approach* *Absence of Omission Hazardous Failure Mode* *Absence of Commission Hazardous Failure Mode* *Absence of Early Hazardous Failure Mode* *Absence of Late Hazardous Failure Mode* *Absence of Value Hazardous Failure Mode* *Effects of Other Components* *Handling of Software Failure Mode* *Handling of Hardware/Other Component Failure Mode*