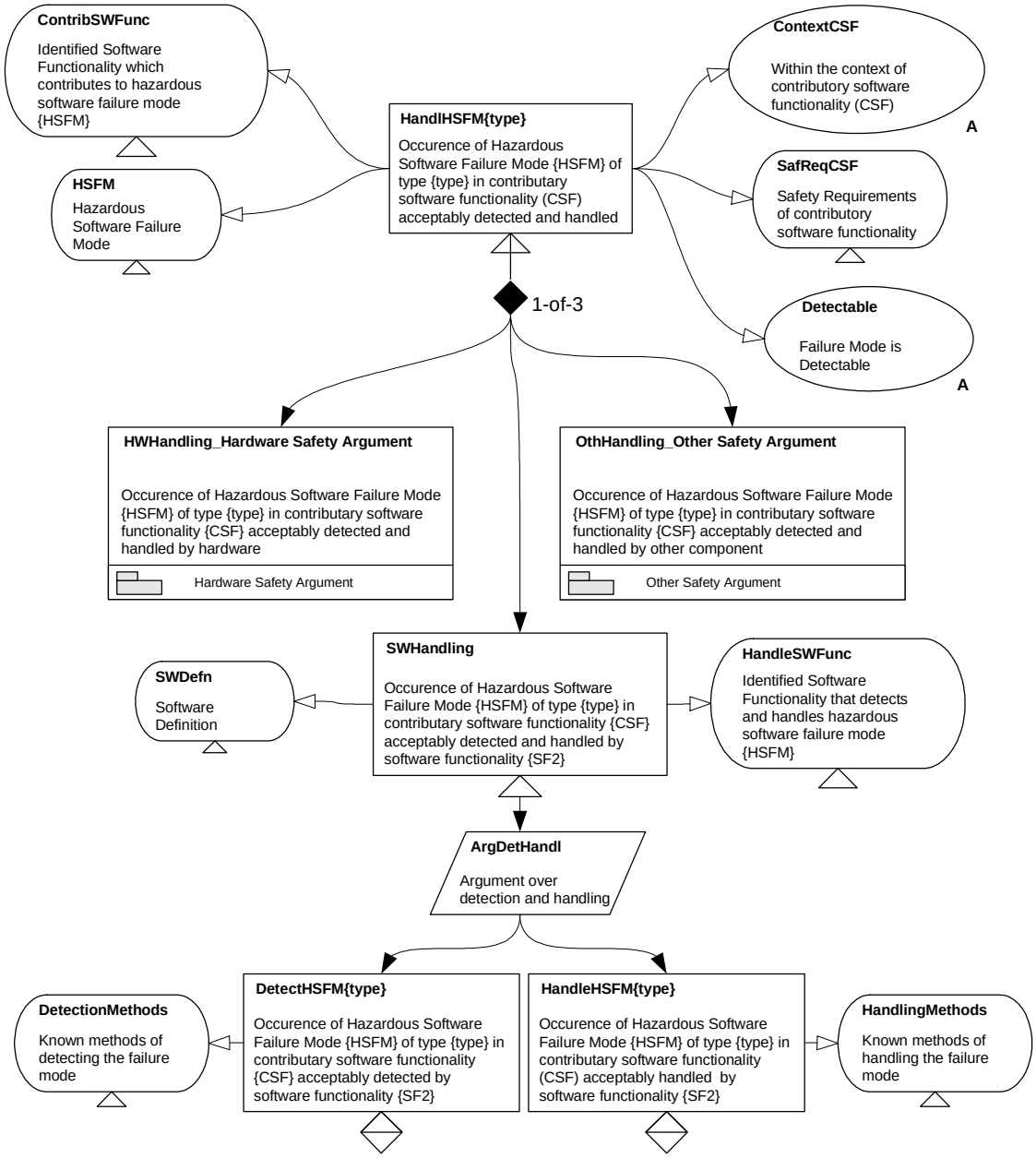
Handling of Software Failure Mode

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 develop an argument that a software failure mode can be handled by other components (software, hardware or other).
Also Known As	
Motivation	The motivation for this pattern is to be able to either identify requirements on the hardware or other component safety arguments, or to develop an argument about other software functionality that will detect and handle the failure.

Structure



Participants

HandlHSFM {type}	The overall objective of the argument - to provide sufficient support for the claim that that a particular type of Software failure mode can be handled by another component.
HSFM	This context identifies the Hazardous Software Failure Mode, for which this pattern develops the handling argument.
ContribSWFunc	This context describes the software functionality that has a contributing effect to the cause of the software failure mode.

	SafReqCSF	The safety requirements of the contributory software functionality are given as a basis for developing evidence.
	ContextCSF	An assumption is made that only circumstances in which the Contributory Software Functionality (CSF) operates are considered during analysis of the failure mode.
	Detectable	This argument assumes that the software failure mode is detectable. A handling argument cannot be generated for an undetectable failure mode.
	HWHandling_ Hardware Safety Argument	This away goal to the hardware safety argument places a requirement on that argument that the Failure Mode can be handled by the hardware
	OthHandling_ Other Safety Argument	This away goal to the other component safety argument places a requirement on that argument that the Failure Mode can be handled by the other component
	SWHandling	This claim asserts that the software failure mode can be handled by another piece of software functionality
	SWDefn	This Software Definition should give a clear description of the system software. From the model it should be possible to determine how the software functionality can handle the failure of the contributory software functionality.
	HandlSWFunc	This context describes the software functionality that can detect and handle the occurrence of the software failure mode.
	ArgDetHandl	This strategy describes the argument approach – decomposing across the detection and the handling of the software failure mode.
	DetectHSFM {type}	This claim asserts that the software failure mode can be detected by the other software functionality

	DetectionMethods	This context provides the possible detection methods based upon the type of the failure mode. Detection methods include: Omission: Detection on Time (infinite threshold) Commission: Detection on Time (early) and/or unexpected input Early: Detection on Time (Early) Late: Detection on Time (Late) Coarse Value: Detection on out of safe bounds (e.g. range, rate of change) Subtle value failures can be detected if redundancy is employed.
	HandleHSFM {type}	This claim asserts that the software failure mode can be handled by the other software functionality
	HandlingMethods	This context provides the possible handling methods based upon the type of the failure mode and whether redundancy is employed.
Collaborations	• **SWDefn**, **ContribSWFunc**, **HSFM** should be suitable for identifying the handling software functionality identified in **HandleSWFunc**. • **DetectionMethods** should be suitable for identifying the argument below **DetectHSFM{type}**. • **HandlingMethods** should be suitable for identifying the argument below **HandleHSFM{type}**.	
Applicability	This pattern identifies the claims about handling a software failure mode by parts of the system (hardware, other software functionality, other components) for a particular software failure mode. It assumes that the failure mode has been identified, classified as a certain type and the Contributory Software Functionality has been identified. It also assumes the ability of the other parts of the system to handle the software failure mode can be identified. The pattern is only applicable to failure modes that can be detected. Undetectable failure modes cannot be argued about using this pattern.	
Consequences	After instantiating this pattern the following undeveloped goals may remain:	

	• **DetectHSFM{type}** and **HandleHSFM{type}** After instantiating this pattern one of two away goals may need to be satisfied: • **HWHandling_Hardware Safety Argument**, **OthHandling_Hardware Safety Argument** To satisfy the decomposition of **HandlHSFM{type}** the necessary goals need to be decomposed and the away goals satisfied.
Implementation	This pattern should be instantiated in a Top Down fashion. All goals and contexts should be instantiated before continuing to a lower level in the pattern. It should be determined whether the failure mode is detectable before trying to decompose the argument. A choice (1-of-3) must be made about whether the handling of the software failure mode is provided by hardware, other software functionality or another component. **Possible Pitfalls** • Not correctly determining whether the failure mode is detectable or undetectable can lead to an argument being generated that does not cover the failure mode in all possible contexts. • Not correctly identifying the correct detection or handling approaches for the failure mode can lead to an incorrect argument being developed. • Providing a requirement on the hardware or other component safety argument, which cannot be supported.
Examples	None provided at this stage.
Known Uses	
Related Patterns	*Software Argument Approach* – This pattern has an undeveloped goal which can be the overall objective of *Handling of Software Failure Mode.* 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*