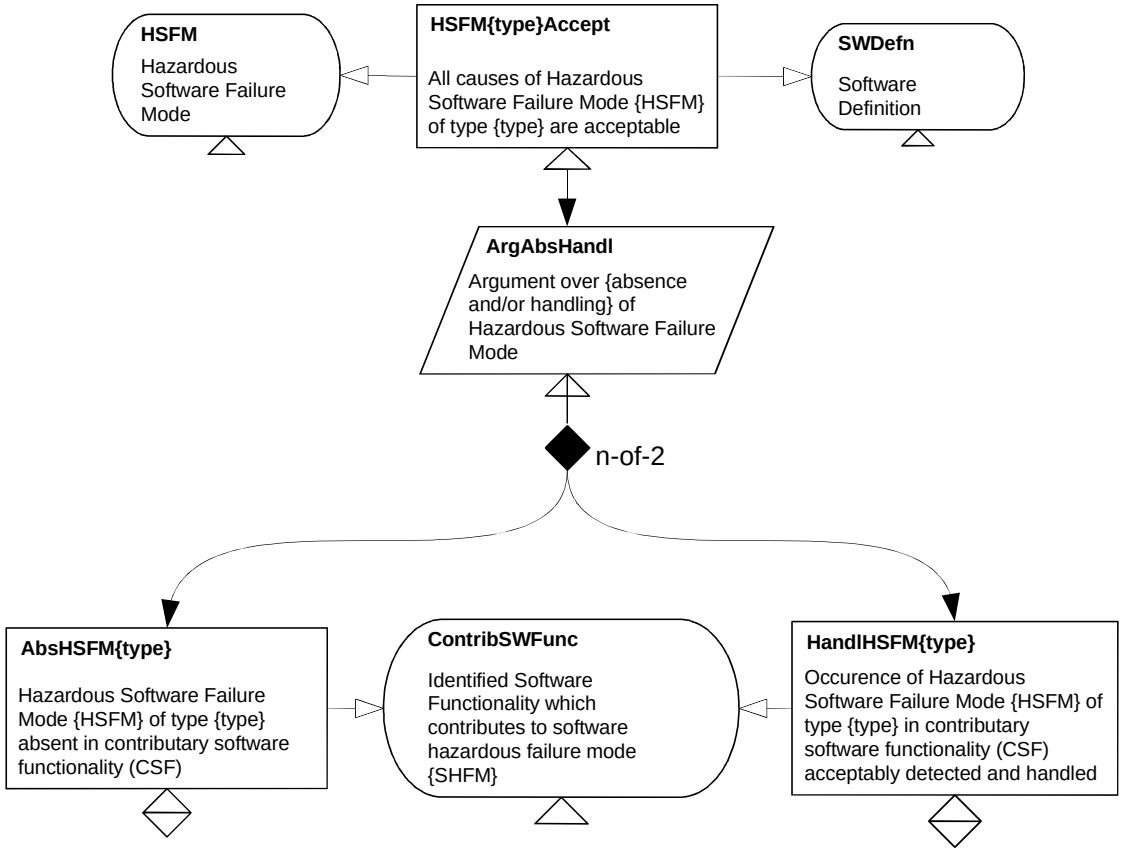
Software Argument Approach

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 identify the argument approach used for demonstrating the acceptability of the hazardous software failure mode. The argument can be made by showing Absence and/or Handling of the failure mode.
Also Known As	
Motivation	Arguments for the acceptably safe nature of a hazardous software failure mode can be made two ways. As it is not possible to determine a probability for systematic software failures, evidence must be provided that the failure mode is absent or can be handled if it does occur. The structure of the pattern allows for a mixture of both argument approaches, depending upon whether individually or together enough evidence can provided to support the claims.

Structure



Participants

HSFM{type}Accept	The overall objective of the argument – to provide sufficient support to the claim that the Hazardous Software Failure Mode of a particular type under consideration is acceptably safe.
HSFM	This context identifies the Hazardous Software Failure Mode, for which this pattern develops the argument approach.
SWDefn	This Software Definition should give a clear description of the system software. From the model it should be possible to determine the contributory software functionality in which the failure mode is manifested.
ArgAbsProbHandl	This provides the strategy for arguing about the safety of the hazard. The argument can be decomposed by showing absence and/or handling of the failure mode.

	AbsHSFM{type}	This claim asserts that the hazardous software failure mode does not exist in the software, and thus cannot contribute to the hazard occurring.
	ContribSWFunc	This context describes the software functionality that may have a contributing effect to the cause of the software failure mode.
	HandlHSFM{type}	This claim asserts that the hazardous software failure mode occurring in the particular software functionality can be handled through other means.
Collaborations	• The contributory software functionality identified in **ContribSW** should be determined from the software definition (**SWDefn).** • **ContribSW** identifies the software functionality on which the **AbsHSFM{type}** and **HandlHSFM{type}** claims are made. • It is necessary for the goals **AbsHSFM{type}**, **HandlHSFM{type}** to be suitable for providing an argument about the acceptability of the failure mode.	
Applicability	This pattern identifies the argument approach for a particular software failure mode. It assumes that the failure mode has been identified and classified as a certain type. It also assumes that evidence can be generated about the absence or handling of the failure mode.	
Consequences	After instantiating this pattern one or two undeveloped goals remain: • **AbsHSFM{type}** &/or **HandlHSFM{type}** The above goal(s) must be developed to satisfy the decomposition of **ArgAbsHandl**.	
Implementation	To instantiate this pattern the means by which the argument is going to be satisfied should be chosen. The choice of the two claims (Absence and/or Handling) is an m of n selection. It is up to the implementer to choose what technique(s) will be used, depending upon the detail of the failure mode. Where sufficient evidence cannot be generated about absence or handling of the failure mode alone, it is recommended that a combination of these two types of evidence is used.	

	Possible Pitfalls • Selecting an argument approach for which evidence cannot be generated.
Examples	None provided at this stage.
Known Uses	
Related Patterns	*Hazardous Software Failure Mode Classification* – This pattern provides a context for the overall objective **HSFM{type}Accept**. *Absence of Omission Hazardous Failure Mode* – This pattern can be used to decompose the undeveloped goal **AbsHSFM{type}** for a failure mode of type Omission. *Absence of Commission Hazardous Failure Mode* – This pattern can be used to decompose the undeveloped goal **AbsHSFM{type}** for a failure mode of type Commission. *Absence of Early Hazardous Failure Mode* – This pattern can be used to decompose the undeveloped goal **AbsHSFM{type}** for a failure mode of type Early. *Absence of Late Hazardous Failure Mode* – This pattern can be used to decompose the undeveloped goal **AbsHSFM{type}** for a failure mode of type Late. *Absence of Value Hazardous Failure Mode* – This pattern can be used to decompose the undeveloped goal **AbsHSFM{type}** for a failure mode of type Value. *Handling of Software Failure Mode* – This pattern can be used to decompose the undeveloped goal **HandlHSFM{type}**. 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*