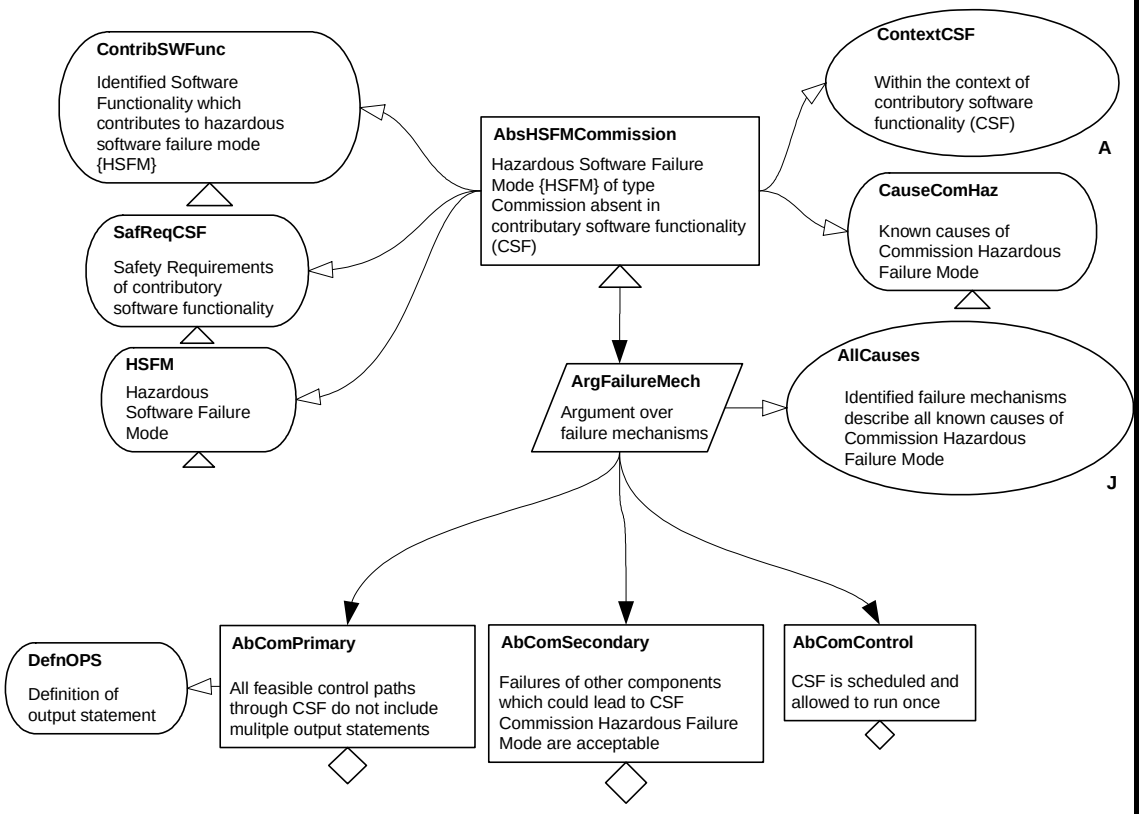
Absence of Commission Hazardous Failure Mode

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 argue that an individual hazardous software failure mode, which is of the type *Commission*, is absent within a certain component of software functionality in a system.
Also Known As	
Motivation	The motivation for this pattern is to be able to decompose the possible causes of a Commission Failure Mode so as to be able to analyse them individually. These causes are based upon the software in which the failure mode manifests itself, other components of the system that may induce the failure mode within the software functionality and control of the software functionality.

Structure



Participants	**AbsHSFM Commission**	The overall objective of the argument - to provide sufficient support for the claim that all possible causes of a particular Commission Failure Mode are absent within the software that could contribute towards such a failure mode.
	ContribSWFunc	This context describes the software functionality that may have 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.
	HSFM	This context identifies the Hazardous Software Failure Mode, for which this pattern develops the argument.
	ContextCSF	An assumption is made that only circumstances in which the Contributory Software Functionality (CSF) operates are considered during analysis of the failure mode.
	CauseComHaz	To develop the argument it is necessary to take into account all possible causes of a Commission failure mode and these are made explicit within this context.
	ArgFailureMech	This strategy describes the argument approach – decomposing across all known failure mechanisms
	AllCause	The justification of the argument given is that all possible causes shown in **CauseComHaz** are explicitly developed in the argument that follows.
	AbComPrimary	This claim is associated with primary failures within the Contributory Software Functionality (CSF). The claim encompasses the potential commission failure caused by the CSF by setting the goal that there should not be multiple "output statements".
	DefnOPS	This context gives a clear definition of what is meant by an output statement within the CSF.

	AbComSecondary	This claim is associated with secondary failures relating to other components within the system on which the CSF is dependent. The goal states that failures in other components of the system, which could lead to the Contributory Software Functionality producing a failure mode, are acceptable.
	AbComControl	This claim is associated with the items with control over the CSF, namely the scheduling.
Collaborations	• **CauseProvHaz** identifies the three claims that must be met: **AbComPrimary**, **AbComSecondary** and **AbComControl**. The Justification **AllCause** distinguishes that these claims cover all possible causes of the hazardous failure mode • **ContribSWFunc** identifies the contributory software functionality on which the **AbComPrimary** claim is made.	
Applicability	This pattern identifies the claims about different parts of the system for an argument for particular software failure mode. It assumes that the failure mode has been identified, classified as a certain type and the Contributory Software Functionality can be identified. It also assumes that evidence can be generated about the absence of the failure mode.	
Consequences	After instantiating this pattern three undeveloped goals will remain: • **AbComPrimary**, **AbComSecondary** and **AbComControl** To satisfy the decomposition of **AbsHSFMCommission** these three goals need to be decomposed.	
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. This pattern assumes that the software failure mode has already been identified and that it is possible to identify the contributory software functionality. The pattern assumes that it is possible to develop evidence about the contributory software functionality in great enough detail. Any further failure mechanisms other than primary, secondary and control should be identified and added to the decomposition of the argument. Any additional ways in which the primary or control could cause	

	the failure mode should be added to the decomposition of the argument. **Possible Pitfalls** • Not correctly identifying the Contributory Software Functionality in **ContribSWFunc** may lead to incorrect or insufficient evidence being developed about the absence of the failure mode. • Not correctly considering the context for the use of the Contributory Software Functionality in **ContextCSF** may lead to incorrect or insufficient evidence being developed about the absence of the failure mode. • Not establishing all the possible causes of the failure mode in **CauseComHaz** may lead to incomplete evidence being developed about the absence of the failure mode.
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 *Absence of Commission Hazardous Failure Mode.* *Effects of Other Components*– This pattern can be used to decompose the undeveloped goal **AbOmSecondary**. 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*