CRAM (**C**ode **R**eview ch**A**nges **M**odel)

	Categories not adopted from Beller's Taxonomy			
Elicited in the inception phase of the study	Textual Defects	Debug Info	Problems with debugging messages. Debug Info is placed in this sub-group because it improves programs static and runtime documentation	not applicable
Integrated from the taxonomy of Beller et al.	Textual Defects	Other Textual Defects	Other Textual Defects that could not be placed to other defect classes.	considered in Licensec Header and Typos
Elicited from feedback in the survey	Supported By Language Defects	Element Type	The software element is with wrong type (only cases not causing runtime failure)	not applicable
	Supported By Language Defects	Void Parameter	Using empty brackets instead of keyword "void" as parameter	not applicable
	Supported By Language Defects	Element Reference	Referring to software element with incomplete name	considered in Naming

Artifact	Activity	Category	Topic	Detailed Change
Production & Test Code	**Maintainability / Perfective Maintenance** (Modification of a software product after delivery to improve performance or maintainability)	**Documentation**	**Textual Documentation** Issues concerning the documentation through textual representation, such as naming of classes, method, variables. This also includes license headers, typos in either line comments or Javadoc	**Naming** Problems relating to software element (e.g., methods, classes, variables, etc) names that do not conform to the naming policy of the project
				Comments Explanations of complex code fragments, classes, methods. Issues include wrongly placed comments, missing comments, missing or wrong Javadoc etc.
				License Header Issues regarding missing or wrong license headers inside source-files
				Typos Spelling mistakes in the documentation
			Language Supported Documentation Documentation through statements/elements that the programming language offers (e.g., java public modifier to document that it is accessible from the outside)	**Immutability** Not declaring a variable to be immutable when it should have been or declaring it immutable when it should have not been
				Visibility (Modifiers) Software element (e.g. method, variable, class) has too much or too restricted visibility
		Style		**Brackets & Braces** e.g., single statement after a conditional branch
				Indentation consistent indentation of the code
				Blank Lines excess of blank lines or too few blank lines or wrong split of lines
				Long Lines code statement too long, over a specific amount of characters
				Whitespace Usage usages of blank spaces in the code
				Grouping grouping of methods with related functionality or adding class variables at the beginning of the class
				Commented out code remove code that is commented out (also TODO and FIXME)
		Structure	**Re-implementation** Structural defects require an alternative implementation method. For example, replacing the program's array data structure with a vector and knowing the existence of prebuilt functionality that could be used instead of a self-programmed implementation would be considered a solution approach defect. Therefore, solution approach defects are not about re-organizing existing code but rethinking the current solution and implementing it in a different way.	**Semantic Duplication** Code structures that have a similar intention but are implemented syntactically different
				Semantic Dead Code Code fragments that are executed, but they do not serve any meaningful purpose and/or have no effect on the result
				Change Function Change function call to another function because it uses old or deprecated functions
				Standard Coding Conventions Use exceptions for error messaging instead of return values, use predefined constants instead of magic numbers, built-in data structures instead of own implementation etc.
				New Functionality new functionality to ensure evolvability, e.g., create new classes, methods to make code more maintainable
				Strings (Wording) Issues regarding contents of strings, badly composed strings
				Logging Add the ability to methods for logging results or errors
				Testing Issues regarding test coverage, wrong tests, additional tests etc.
			Organization Defects that can be fixed by applying structural modifications to the software. Moving a piece of functionality from module A to module B is a possible strategy for this.	**Imports** Issues with wrong or missing or unused import statements
				Move Functionality move functions, part of functions, or other functional elements to a different class, file, or module
				Long Sub Routine split long and complex functions into multiple functions
				Dead Code remove code that is never reached and executed
				Duplication / Redundant Code remove duplicate code or code that is not used
				Complex Code / Simplification restructure or rewrite implementation to make it more understandable
				Statement Issue splitting, combining or otherwise reorganizing a statement inside a function
				Consistency Means the need to keep code consistent in a sense that similar code elements operate in a similar fashion and are more or less symmetrical. For example, similar tasks in similar classes should have similar implementations
				Architectural changes code reviews often result in a change to the system architecture, like splitting an interface into two distinct interfaces, introducing abstractions, or the inclusion of design patterns
	Functionality / Corrective Maintenance (Reactive modification of a software product performed after delivery to correct discovered problems.)	**Interface** Communication with a different part of the system		**Function Call** call to another part of system or libary is incorrect or missing
				Parameter function call or other interaction has incorrect or missing parameters
		Logic		**Compare** mistake in a comparison statement
				Computation computations produce incorrect results
				Wrong Location correct operation is performed, but it is done too soon or too late
				Algorithm/Performance inefficient algorithm is used
		Resource		**Variable Initialization** Variables are left uninitialized prior to use. Uninitialized variables may contain any value and using such variable for comparison or calculation produces arbitrary results.
				Memory Management Mistake is made in handling the system memory.
				Data & Resource Manipulation Defects related to manipulating or releasing data or other resources.
				Security Issues related to the application's/software's security aspects
				Concurrency Issues regarding concurrency
		Check		**Check Function** when a function is called there is also a need to check that the value returned is valid and that no error occurred
				Check Variable there is a need to check variable
				Check User Input the need to validate user input
		Larger Defects		**Completeness** partially implemented feature
				GUI Defects in the user interface code relating to the consistency of the user-interface, and to the options made possible to the user in each situation.
				Check outside code / Domino Effects Defects that required that part of the application code that was not under review to be checked, as it was likely to contain incorrect code based on the current review.

Other Changes Changes not typically found in source-code files (.java, .py, .cpp etc.) which are nonetheless essential to the runtime of a project	**Commit Message** Updates/changes in the commit message of a submitted patch. Mostly related to wrong description of the change or not capturing all changes.
	Continuous Integration / Continuous Deployment configurations Changes to configuration files concerning the Continuous Integration or Continuous Deployment pipeline/setup.
	Automated Static Analysis Tools configurations Changes in the configuration of Linters, Checkers, Recommenders used in the project (e.g., Checkstyle, PMD, FindBugs etc.)
	Language or Framework specific Changes to files native to the used programming language. For example MANIFEST for Java.
	External Software Documentation Changes to the external Software Documentation files
	Runtime Configurations docker-configs, ansible playbooks, deployment configs etc.
	Other Includes changes to XML, Scripts, README files, HTML files and Version Control