Table 4: Definition and Frequency of locations, reasons, and purposes

	Category	Definition	Frequency	
Location	**Externals**	Build code that configures factors that are external to the projects. These factors include platforms both hardware and software, tools that are used in the build execution, build plugin features/libraries/dependencies for building the project, and version of any of these artifacts (including build systems).	**115**	**(58%)**
	Platform configuration		45	(23%)
	Tool configuration		30	(15%)
	Libraries and plugins		28	(14%)
	Artifact versioning		12	(6%)
	Behavioural	Build code that configures the behavior of the build system. Such as environmental variables and flags that alter the commands, build variables or override inherited variables that are used in the build execution, project descriptive information, the configuration of avoids redundancies or duplicate configurations through inheritance, passive behavior that does not affect artifacts produced by the build system.	**61**	**(30%)**
	Dynamic settings		35	(18%)
	Build variables		20	(10%)
	Project metadata		3	(1%)
	Multi-directory configuration		2	(1%)
	Logging		1	(1%)
	File System	Build code that specifics file system on a logical or physical layer.	**24**	**12**%
	Logical file system		15	(8%)
	Physical file system		9	(5%)
Reason	**Limitation**	Constraints imposed by the design or implementation of third-party libraries or development tools.	**75**	**(38%)**
	External tool limitation		32	(16%)
	External library limitation		22	(11%)
	Build tool limitation		21	(11%)
	Configuration	Configuration issues during the compilation and build process, such as compiler configuration, symbol visibility, file path styles in different platforms, and checking existence of files/features/libraries/dependencies.	**74**	**(37%)**
	Compiler configuration		31	(16%)
	Symbol visibility		22	(11%)
	Platform-specific setting		11	(6%)
	Feature existence		10	(5%)
	Dependency	Dependency issues due to unavailable artifacts or assets, such as missing stale dependencies, management of internal dependencies, and dependency conflicts.	**13**	**(7%)**
	Missing dependency		7	(4%)
	Internal dependency management		5	(3%)
	Dependency conflict		1	(1%)
	Code smell	Violations of fundamentals of design principles, i.e., instances of poor coding practice in build files.	**12**	**(6%)**
	Recursive call	Coherence issues, recursive calls to invoke another build file.	**6**	**(3%)**
	Document	Inadequate project description issues, such as licensing and metadata specification.	**5**	**(3%)**
	Specify metadata		4	(2%)
	Licensing		1	(1%)
	Release and install behaviors	Sanitize project before releasing or post-install files or program after building	**3**	**(2%)**
	Release		2	(1%)
	Post-install		1	(1%)
	No reason	A label could not be assigned (due to lack of information).	**12**	**(6%)**
Purpose	Document for later fix	Document an issue that should be revisited in the future.	68	(34%)
	Warning for future developers	Warn other developers to pay attention to an aspect of the solution that may not be clear from its structure or content.	52	(26%)
	Document suboptimal implementation choice	Explain why a problematic solution has been adopted.	46	(23%)
	Document workaround	Explicitly document constraints imposed by design or implementation choices. The comment contains workaround-related keywords, such as "workaround" and "temporary".	18	(9%)
	Placeholder for later extension	Document an extension point for later enhancement(s).	12	(6%)
	Silence build warnings	Defer or ignore warnings emitted by underlying tools.	4	(2%)