Dimension	Metrics	Values	Definition (d) – Rationale (r)
Code Metrics	Log revision count	Numeric	d: The number of commits prior which had lo statement changes.
			r: This helps to identify if the file is prune to log statement changes.
	New File	Boolean (0 -1)	d: Check if the log is added in a new file (i.e., newly comitted)
			r: This helps to identify which log statements where added later in subsequent commits fron the inital commit logs
	Total Revision Count	Numeric	d: Total number of commits made to the file since log statement is added.
			r: This helps to find out if the file is changed heavily which can result in log changes (cite paper on after thoughts)
	Code churn in commit	Numeric	d: The code churn of the commit in which log is added.
			r: Log changes are correlated to code churn in files. (Ian's paper EMSE)
	Variables declared	Numeric	d: The number of variables which are declared before the log statement. (we limit to 20 lines before log statement)
			r: When new variables are declared, developer may log the new variables to obtain more information (Afterthouhgs paper cite)
	SLOC	Numeric	d: The number of lines of code in the file.
			r: Large files have more functionality and are more prune to changes (paper3) and more lo changes. (after thoughts and field debugging EMSE)
Log Metrics	Log Context	Categorical	d: Identify the block in which log statement is added. (i.e., 'if', 'if-else', 'try-catch', 'exception', 'throw', 'new function')
			r: Prior research find that logs are used in assertion checks, logical brancing, return valu checking, assertion checking (Where do developers log)
	Log Level	Categorical	d: Identify the log level (verbosity) of the added log. (i.e., 'info', 'error', 'warn', 'debug' 'trace' and 'trace')
			r: Developers spend signifcant amount of time in adjusting the verboisty of logging statemetns (Characteriging logs)
	Log variable count	Numerical	d: Number of variables logged.
			r: Over 62% of logging statemetns end adding new variables (Characteriging logs) . Hence fewer variables in inital log statement might result in addition later.
	Log text length	Numerical	d: Number of text phrases logged (i.e., we count all text present between a pair of colons as one phrase)
			r: Over 45% of logging statemetns have modifications to static context (Characterigin logs). Logs with fewer phrases might be subject to changes later to provide better explanation
	Log density	Numerical	d: Ratio of number of log lines to the source code lines in the file.
			r: Research has found that there is one log lin per 30 lines of code. (Charaterzing logs). If it

Dimension	Metrics	Values	Definition (d) – Rationale (r)
Developer Metrics	Resolution time	Numerical	d: The time it takes for the issue to get fixed. It is defined as the time it takes since an issue is opened till its closed.
			r: More resolution time might suggest more complex fix with more code churn resulting in more log churn.
	Number of developers involved	Numeric	d: Total number of unique developers who comment on the issue report on JIRA
			r: Components with many unique authors likely lack strong ownership, which in turn may lead to more defects (paper4) and change logging statemetns (EMSE Ian).
	Number of Comments	Numeric	d: Total number of discussion posts on the issue.
			r: Number of comments is correlated to the resolution time of issue reports (Predicting the fix time of bugs. In RSSE Giger). More comments may also indicate the issue is more complex requiring more code churn and logging statement changes.
	Devloper experience	Numeric	d: The number of commits the developer has made prior this commit.
			r: Research has shown that experienced developers might take up more complex issues(Ownership, experience and defects: a fine-grained study of authorship) and therefore may leverage logging statemetns more (EMSE IAN).
	Issue type	Categorical	d: Identify the type of issue i.e., 'Bug', 'Improvement', 'Task', 'New Feature', 'Sub-Task', 'Test'
			r: Some issue types might have higher code churn than others (example: Bug and New features might have more code churn when compared to Sub-Tasks) and are committed faster.
	Priority type	Categorical	d: Identigy the priority of the issue i.e., 'Critical', 'Blocker', 'Major', 'Minor' and 'Trivial'
			r: Research has shown that priority of issue effects resolution time of bug fixes (Studying the Fix-Time for Bugs in Large Open Source Projects). Higher the priorityindicates the issue will be fixed faster with logging statement changes.

Table 2: