Table 1: Description of TravisTorrent's data fields and one sample data point from rails/rails

Column Name	Description	Unit	Example
`row`	Unique identifier for a build job in TravisTorrent	Integer	1543966
`git_commit`	SHA1 Hash of the commit which triggered this build (should be unique world-wide)	String	c1d9c11cbe3d20f2...
`git_merged_with`	If this commit sits on a Pull Request (`gh_is_pr true`), the SHA1 of the commit that merged said pull request	String	
`git_branch`	Branch git_commit was committed on	String	4-1-stable
`git_commits`	Preceding commits that were not built (e.g., transferred in one push, ...) this build comprises	List of Strings	87a2f02199d21a2aa...
`git_num_commits`	The number of commits in `git_commits`, to ease efficient splitting	String	1
`git_num_committers`	Number of people who committed to this project	Integer	1
`gh_project_name`	Project name on GitHub (in format user/repository)	String	rails/rails
`gh_is_pr`	Whether this build was triggered as part of a pull request on GitHub	Boolean	false
`gh_lang`	Dominant repository language, according to GitHub	String	ruby
`gh_first_commit_created_at`	Push date of first commit in `git_commits` to GitHub	ISO Date (UTC+1)	2014-04-18 20:12:32
`gh_team_size`	Size of the team contributing to this project	Integer	168
`gh_num_issue_comments`	If `git_commit` is linked to an issue on GitHub, the number of comments on that issue	Integer	0
`gh_num_commit_comments`	The number of comments on `git_commit` on GitHub	Integer	0
`gh_num_pr_comments`	If gh_is_pr is true, the number of comments on this pull request on GitHub	Integer	0
`gh_src_churn`	The churn of `git_commit`, i.e. how much production code changed in the commit, based on lines	Integer	4
`gh_test_churn`	The churn of `git_commit`, i.e. how much test code changed in the commit, based on lines	Integer	8
`gh_files_added`	Number of files added in `git_commit` (this is generally correlated with the churn)	Integer	0
`gh_files_deleted`	Number of files deleted in `git_commit` (this is generally correlated with the churn)	Integer	0
`gh_files_modified`	Number of files modified in `git_commit` (this is generally correlated with the churn)	Integer	3
`gh_tests_added`	How many test cases were added in `git_commit` (e.g., for Java, this is the number of @Test annotations)	Integer	0
`gh_tests_deleted`	How many tests were deleted in `git_commit` (e.g., for Java, this is the number of @Test annotations)	Integer	0
`gh_src_files`	Number of production files in the repository	Integer	
`gh_doc_files`	Number of documentation files in the repository	Integer	
`gh_other_files`	Number of remaining files which are neither production code nor documentation	Integer	
`gh_commits_on_files_touched`	Number of commits that touched (added/deleted/modified) the files in git_commit previously	Integer	93
`gh_sloc`	Number of executable production source lines of code, in the entire repository	Integer	53421
`gh_test_lines_per_kloc`	Test density. Number of lines in test cases per 1,000 `gh_sloc`	Double	2191.011
`gh_test_cases_per_kloc`	Test density. Number of test cases per 1,000 `gh_sloc`	Double	188.3342
`gh_asserts_cases_per_kloc`	Assert density. Number of assertions per 1,000 `gh_sloc`	Double	535.0143
`gh_by_core_team_member`	Whether this commit was authored by a core team member	Boolean	true
`gh_description_complexity`	If `gh_is_pr` is true, the Pull Request's textual description complexity	Integer	
`gh_pull_req_num`	Pull request number on GitHub	Integer	
`tr_build_id`	Unique build ID on Travis	String	23298954
`tr_status`	Build status (pass, fail, errored, canceled)	String	passed
`tr_duration`	Overall duration of the build	Integer (in seconds)	23389
`tr_started_at`	Start of the build process	ISO Date (UTC)	2014-04-18 19:12:32
`tr_jobs`	Which Travis jobs executed this build (number of integration environments)	List of Strings	[23298955, ...]
`tr_build_number`	Build number in the project	Integer	15459
`tr_job_id`	This build job's id, one of `tr_jobs`	String	23298981
`tr_lan`	Language of the build, as recognized by BuildLogAnalyzer	String	ruby
`tr_setup_time`	Setup time for the Travis build to start	Integer (in seconds)	0
`tr_analyzer`	Build log analyzer that took over (ruby, java-ant, java-maven,java-gradle)	String	ruby
`tr_frameworks`	Test frameworks that `tr_analyzer` recognizes and invokes (junit, rspec, cucumber, ...)	List of Strings	testunit
`tr_tests_ok`	If available (depends on `tr_frameworks` and `tr_analyzer`): Number of tests passed	Integer	310
`tr_tests_fail`	If available (depends on `tr_frameworks` and `tr_analyzer`): Number of tests failed	Integer	1
`tr_tests_run`	If available (depends on `tr_frameworks` and `tr_analyzer`): Number of tests were run as part of this build	Integer	311
`tr_tests_skipped`	If available (depends on `tr_frameworks` and `tr_analyzer`): Number of tests were skipped or ignored in the build	Integer	
`tr_failed_tests`	All tests that failed in this build	List of strings	SerializedAttributeTest
`tr_testduration`	Time it took to run the tests	Double (in seconds)	28.2
`tr_purebuildduration`	Time it took to run the build (without Travis scheduling and provisioning the build)	Double (in seconds)	
`tr_tests_ran`	Whether tests ran in this build	Boolean	true
`tr_tests_failed`	Whether tests failed in this build	Boolean	true
`tr_num_jobs`	How many jobs does this build have (length of tr_jobs)	Integer	30
`tr_prev_build`	Serialized link to the previous build, by giving its `tr_build_id`	String	39557888
`tr_ci_latency`	Latency induced by Travis (scheduling, build pick-up, ...)	Integer (in seconds)	1408

Example 1: Standard output from Maven regarding tests

```
1  -------------------------------------------------------
2   T E S T S
3  -------------------------------------------------------
4  Running nl.tudelft.watchdog.ClientVersionCheckerTest
5  Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time
     elapsed: 0.04 sec
6
7  Results :
8
9  Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
10
11 [INFO] All tests passed!
```

Example 1 shows an excerpt of one test execution from the TestRoots/WatchDog project. In the output, we can see the executed test classes (line 4), and how many tests passed, failed, errored and were skipped. We also get the test execution time (line 5). Moreover, Maven prints an overall result summary (line 9) that the Buildlog Analyzer uses to triage its prior findings. Line 11 shows the overall test execution result. Our Buildlog Analyzer gathers all this information and creates, for each invoked project, a CSV table with all build and test results for each job built. We then aggregate this information with information from the build status analyzer step by joining their output. TravisTorrent provides convenient access to this data. Gradle is much less verbose than Maven, providing us with fewer information.

By contrast, in Ruby, the test framework is responsible for the console output: it is no different to invoke RSpec through Rake than through Bundler, the two predominant Ruby build tools. For Ruby, we support the prevalent Test::Unit and all its off springs, like MiniTest. Moreover, we capture behavior driven tests via RSpec and Cucumber support.

B.3 Data Linearization And Synthesis

If we want to answer questions such as "Does the use of CI lead to higher-quality products?", we need to make a connection between the builds performed on Travis CI and the repository which contains the commits that triggered the build. We call this build linearization and commit mapping, as we need to interpret the builds on Travis CI as a directed graph and establish a child-parent relationship based on the git commits that triggered their execution. Although this sounds trivial, since there should be a 1:1 relationship between builds and commits, there are six different scenarios (a–f) arising from git's non-linear nature that we discuss in the following. During this step, we also assessed the status of the project at the moment each build was triggered by extracting and synthesizing information from two sources: the project's git repository and its corresponding entry in the GHTorrent database.

Figure 2 exemplifies a typical GitHub project that uses Travis CI for its CI. In the upper part ①, we see the Travis CI builds (§1-