Metrics in Agile and Rapid Software Development

López, L., Burgués, X., Martínez-Fernández, S., Vollmer, A.M., Behutiye, W., Karhapää, P., Franch, X., Rodríguez, P., & Oivo, M. Quality Measurement in Agile and Rapid Software Development: A Systematic Mapping [Dataset], v.1. Universitat Politècnica de Catalunya.

Quality Requirements Metrics (ISO/IEC 25010)

QR (#metrics)	*Metrics as reported in primary studies*
Reliability (41)	Defect modifications; defect density; number of defects carried over the next iteration; list of open defects; mean time to failure; mean time before failure; passed tests; fast tests; test coverage (testing status); bug density; non-bug density; errors at runtime; availability uptime; postponed issues ratio; critical issues ratio; build stability; unit test coverage for the developed code; open defect severity index; allowed error count per time; ServiceInstances.failOverTime; Errors.severe; component failure; error detection; state coverage; transition coverage; path coverage; number of defects; type of defects; number of bugs per time unit; defect density (released defects/KLOEC of code four months after release) found by the customer after release; defect density test defects/KLOEC of code; number of test cases; test coverage; test density per class; number of bugs; fault convergence rate; C1 coverage; number of tested classes; mean value function of detected faults; fault content function; failure detection rate function. *[Note: KLOEC = Thousands of lines of executable code]*.
Maintainability (27)	Complexity; comments; duplication; non-blocking files; well defined issues; total LOC; change rate of the source code; maintainability; highly changed files; weighted methods of a class; coupling between objects; response for a class; lack of cohesion in methods; depth of inheritance tree; number of children; technical debt; access of foreign data; class-level cyclomatic complexity; tight class cohesion; efferent coupling; McCabe's cyclomatic complexity of a function; effective complexity of a file; effective cyclomatic complexity percentage of a file; code incorrectness detectability; lines of code; unit tests; File.size.
Performance efficiency (22)	System performance; maximum time permitted for deployment of new instances; execution; ServiceInstances.count; operational performance; average CPU usage; instantaneous CPU usage; memory usage; latency; performance response time; number of transactions per second; response time; transaction processing response time; page rendering response time; query processing response time; number of pages rendered per second; number of messages per second; scene graph size; milliseconds/frame; maximum polygon & object counts at X fps (frame per second); external graphics card usage if the hardware already have one and it's getting used; throughput.
Security (9)	Count of baseline vulnerabilities; count of regressive vulnerabilities; count of new vulnerabilities; enforcement ratio; framework misuse; number of open security defects; probability of successful attack; impact of attack; number of vulnerabilities.
Usability (5)	Screen switch time; visual/instrumentation failure; number of usability issues met by users; importance of usability issues met by users; User satisfaction.
Functional suitability (2)	End user feedback; feature usage (software usage).
Not classified (1)	Lines of code *[because the primary study reported it as a reliability metric but we would map it to maintainability]*.

Quality Management Indicators

QMI (#reported QMI)	*QMIs as reported in primary studies*
Product Quality (35)	Architectural integrity, Architecture stability, Blocking, Blocking code, Code Quality, Continuous Code Quality, Defect rate, Defects delivered to the customer, Development progress, Development Quality, External Quality, Field Quality, Internal Quality, Post-release quality, Pre-release quality, Product quality (based on security), Product Quality, Product Readiness, Product Reliability, Pull Request Release Readiness, Quality, Quality feedback loop, Realized quality requirements, Release Quality, Release Readiness, Software design stability, Software quality, Software Reliability, Software stability (bugs), Software usage, System complexity, Team's maintenance quality, Technical Debt, Testing Status, Value
Productivity (23)	Coding activity, Defect fixing efficiency, Defect fixing effort, Delivery speed, Development productivity, Development progress, Development speed, Development status, Estimated effort, Feature throughput, Fixing speed, issues' velocity, Productivity, Productivity rate, Programmer Productivity, Project cost and time, Project progress, Release frequency, Resource Productivity, Sprint progress, Team Productivity, Testing activities efficiency, Trend development
Risk (7)	Corrected risks, Postponed risks, Relative risk, Risk, Risk (security risk), Risk value, Unhandled risks
Cost (6)	Cost, Development Cost, Project Cost Change, Relative Cost Deviation, Return on Investment (ROI), Technical Debt Cost
Schedule (6)	On-Time Delivery, Overtime, Relative Schedule Deviation, Release Delay, Time to release, Time-to-market
Process Performance (3)	Implementation phase performance, Process performance, System test phase performance
Developer Satisfaction (2)	Developer satisfaction, Team morale
Agility (2)	NERV Agility Index, Activity Agility degree
Project Success (1)	Project Success
Customer Satisfaction (1)	Customer satisfaction

Quality Management Indicators Metrics

QR (#metrics)	*Metrics as reported in primary studies*
Product Quality (122)	#defects = number of open defects at the moment; #defects injected per sprint; #defects/#bugs; (defect_removal_rate) = average number of defects removed during last 4 weeks; (Effort spent for bug fixing / Effort) x 100; (test_execution_rate) = average number of test cases executed during the last 4 weeks; (test_pass_rate) = average number of test cases passed during the last 4 weeks; Amount of programming effort; availability uptime; Bad code smells; Blocking files; Branch Coverage/SLOC; breadth of functionality (most of the functionality, low-functionality); Bug density; Build stability; Code Churn/ReleaseTime (release date and start date of the development phase); code is well structured and commented; Code Quality Checking Rate (Number of builds subject to a code quality check divided by the total number of builds); Code reviews; Coding Violations Density/SLOC; Comment Density/SLOC; commented code; Comments words; Complexity; Critical Issues Ratio; Critical violations; Cyclomatic Complexity; Cyclomatic Complexity/SLOC; Defect density (Defects / KLOC) ; defect density (number of defects per lines of code); "defect modifications (number of identified modifications for each defect per function); defects carried over next iteration; Development task completion; duplicate code; Effectiveness

	0.2*Abstraction + 0.2*Encapsulation + 0.2*Composition + 0.2*Inheritance + 0.2*Polymorphism); Elapsed Frame between Checks(Average number of builds between two builds subject to a code quality check); Elapsed Time between Checks (Average number of days between two builds subject to a code quality check); environment availability; Errors at runtime; Extendibility (0.5*Abstraction–0.5*Coupling+0.5*Inheritance+0.5* Polymorphism); Externally-Visible Quality (released defects/KLOEC of code four months after release) found by the customer after release; Fast test builds; Feature usage; Flexibility (0.25*Encapsulation-0.25*Coupling+0.5*Composition +0.5*Polymorphism); Frequency of program specification change; functional coverage (no metric); Functionality (0.12*Cohesion+0.22*Polymorphism+0.22*Messaging + 0.22*Design Size+0.22*Hierarchies; Heat map revisions per model per week (number of new checked files); Highly changed files: For each commit: files changed, lines of code added/modified/deleted, author, and revision ; Internally-Visible Quality as internal (pre-release) defect density test defects / KLOEC of code KLOEC = Thousands of lines of executable code; Level of programming technologies; Lifetime (open to close pull request); LOC; Median Daily Crash Count; Median Daily Crash Count: the median of the number of crashes per day for a particular version (lower is better); Median Uptime; Median Uptime: the median across the uptime values of all the crashes that are reported for a version (higher is better).; Merge time (open to merge pull request); Most violated rules; Non-bug density; Number of bugs reports; Number of code changes; Number of commit comments; Number of commits; Number of defects detected (overall); Number of defects detected (per pull requests); Number of defects per iteration; number of failed test cases; Number of files added; Number of files deleted; Number of files modified; Number of function parameters; Number of issue comments; number of open defects; Number of participants; number of passed acceptance tests; number of planned/not planned/cancelled test cases; Number of source churn; Number of source files; Number of test churn; Number of trouble reports; Open Defect Severity Index; Pareto distribution of defects per origin; Pass/fall test cases; Passed tests (unit test density, passed/total); Percentage of Checked Branches (Number of branches containing at least one build subject to a code quality check divided by the number of total branches scheduled for build); Percentage of reused modules; Postponed issues ratio; Post-Release Bugs; Post-Release Bugs: the number of bugs reported after the release date of a version (lower is better); Program complexity; Program workload (stress); Relationship of detailed design to requirement; reliability; Requirement analysis; Requirements elicitation and breakdown; response times; Reusability (-0.25*Coupling+0.25*Cohesion+0.5*Messaging +0.5*Design Size); RR= #defects / ((defect _ removal _ rate) − ((test _ execution _ rate) - (test _ pass rate))) ; rules compliance (showing the size); SDI (System Design Instability): percentage of classes whose names change from one iteration to the next + percentage of class which were added + percentage of classes which were deleted; SDIe (System Design Instability with entropy): using the number classes added, deleted, changed and unchanged from the previous iteration; Severity of defects detected; software fix quality (no metric); Source code comments; Specification task completion; status of their code, related to several aspects. E.g. that the code is well structured, that it is easy to read, and that it has low complexity; system resource utilization (such as memory, heap, CPU, disk space, network); technical performance (e.g. speed); Test coverage; Test coverage percentage (overall); Test coverage percentage (per pull request); Test defect density per line of code (defect#/LoC); Test defect density per test scenario (defect#/TS#); Test success (Unit test success density); Testing coverage; Testing effort; throughput rate; TQI (Total quality index) = reusability+Flexibility + Understandability + Functionality + Extendibility + Effectiveness; Types of defects detected; Understandability (-0.33*Abstraction+0.33*Encapsulation-0.33*Coupling+0.33*Cohesion-0.33*Polymorphism–0.33*Complexity–0.33* Design Size); Unit Test Coverage;
Productivity (36)	lines of new code developed per person-day and adjusted for differing levels of product complexity; %effort distribution per project; (Effort spent for bug fixing / Effort) x 100; Bug correction time; Burn-up/down (estimated/done story points); Check in pace; Check-in trend; Code Churn (#files modified in a commit); Delivery on time; Developer Contribution (#commits); developers' effort spent on fixing defects.The number of defects divided by the developers' effort (measured by person hours); Development Time; Fix Time: the duration of the fixing period of a FIXED post-release bug, i.e., the difference between the bug open time and the last modification time (lower value is better);

	Fix Time: the duration of the fixing period of the bug (i.e., the difference between the bug open time and the last modification time). This metric is computed only for bugs with the status FIXED (lower is better).; Fixed Bugs: the number of post-release bugs that are closed with the status field set to FIXED (higher is better); Fixed Bugs: the number of pre-release and post-release bugs that are closed with the status field set to FIXED (higher numbers are better).; integration speed (avg. time to deliver features); Issues completely specified; Mean Complexity; Monthly Active Developers; Number of check-ins current week; Number of check-ins last week; Number of open defects; passed/executed tests; Project M/M (Man * Month); pull requests merged per month (against test coverage and usage of CI for their hypothesis); Rate of New Code (new lines of code); Resolved issues assigned to a date; Schedule Performance Index (SPI): earned value divided by the planned cost measured in person hours; Size/ Effort; Staleness: the number of days the version vi is still in use after a newer version vi+1 has been released; Stories / Person Month Stories in KLOEC (Thousands of lines of executable code); Test progress; Total Lines of Code; Unconfirmed Bugs: the number of post-release bugs with the status field set to UNCONFIRMED; Velocity /Effort;
Process Performance (30)	Average automation duration; Bug Leakage; Bugs ratio; Build performance; CI feedback time; Commit response time; Commit review duration; Commit review iterations; Core component commits; Defect Removal Effort Ratio (person-hour/person-hour); Developer-tester communication; Effort Estimation Capability (person-hour/person-hour); End user feedback; Error correction; Error identification; Fast tests; issue-type in a timeframe; Long tests; Non-issue component commits; Old issues; Productivity (LoC/person-hour); Resolved issues throughput; Team throughput; Test Effectiveness (defect#/person-hour); Test Execution V&V Effectiveness (defect#/defect#); Test Speed (TS#/person-hour); Tests per product; Timely feature delivery; Timely feature specification delivery; Unit test duration
Schedule (13)	% Overtime = (Actual Hours/Expected Hours) -1; ((Real time - Planned time) / Planned time) x 100; Ability to resolve remaining allocated effort; Ability to resolve remaining unallocated effort; Accuracy of planning issues due date: percentage of the difference between the planned due date of past closed issues and the actual tracked issue closing date; actual - planned release date; Core component commits; Non-issue component commits; Percentage of issues larger than the size threshold; Planning accuracy: a percentage of the difference between the planned effort of past closed issues and the actual tracked effort; Ratio of the average past velocity and the theoretical velocity of the available units; Timely feature specification delivery; Timely release delivery
Agility (19)	Agile Process Index; Agile Project Risk Factor; Change Tolerate; Customer Interaction; Flexibility; Free of Modeling and Documentation; Informality; Iterative; People Orientation; Requirements Ambiguity Factor; Requirements Risk Factor; Requirements Volatility Factor; Simplicity; Speed of Execution; Team Communications Index; Team Maturity Index; Team Technical Competency Index; Team Velocity Index; Team-Customer Collaboration Index
Cost (6)	Money; Time; Effort; (1-Cost_pp/Cost_bp) x 100; ((Real costs - Planned costs) / Planned costs) x 100; Value (in percentage terms) delivered per pound spent on development
Risk (5)	Relative_Risk=Effective_M%*NR; Effective MacCabe's cyclomatic Complexity percentage (Effective_M%); Number of revisions of a file (NR); Risk priority number (RPN)=severity ranking, occurrence ranking, detection ranking; probability of successful attack, impact of attack
Project Success (4)	Defect rate (number of errors/complexity); on Budget; on Time; Software size FP = FPUI + FPOU + FPI where FP = Function Point, FPUI = FP based on the number of input / output documents, FPOU = FP based on the number of organizational units, FPI = FP based on the number of interfaces with other modules The result is rounded to the first greater integer value. FPUI = NoUI * WFUI, FPOU= NoOU * WFOU, FPI = NoI * WFI, W = weight
Customer Satisfaction (3)	Customer satisfaction rated between 0-100%, Qualitative data, via interview and customer feedback
Developer Satisfaction (2)	Developer satisfaction rated between 0-100%; Survey question on the Shodan Adherence Survey. The question read, “How often can you say you are enjoying your work?”