

Agile Quality Practices

The worksheet is intended to help teams assess the maturity of their practices and to identify opportunities for improvement. It is not intended for comparison or scoring of teams.



TEAM: _____

DATE: _____

INSTRUCTIONS

- The worksheet should be completed by whole team to ensure consensus within team.
- Each maturity level assumes the lower level practices are already being properly used.

GLOSSARY

INVEST = Stories should exhibit the following features: Independent, Valuable, Estimatable, Small, Testable
 FIRST = Tests should exhibit the following features: Fast, Independent, Repeatable, Small, Transparent
 TDD = Test Driven Development
 BDD = Behaviour Driven Development
 CI = Continuous Integration
 SLA = Service Level Agreement

OBSERVATIONS:

-
-
-

RECOMMENDATIONS:

-
-
-

Level	-1	0	1	2	3	4	5	Observations	Recommendations
Requirements	Features	<ul style="list-style-type: none"> Minimum releasable feature set is NOT defined nor agreed with customer 	<ul style="list-style-type: none"> Minimum releasable feature set (MMR) been identified and agreed with the customer 	<ul style="list-style-type: none"> All the identified features have been prioritised according to business value and estimated in terms of relative effort The features are being continuously reviewed 	<ul style="list-style-type: none"> The Customer / Product Owner define the feature roadmap and are actively involved in prioritisation 	<ul style="list-style-type: none"> All features are linked to business benefits 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community 		
	Stories	<ul style="list-style-type: none"> Requirements are captured using detailed specification documents All requirements have the same priority 	<ul style="list-style-type: none"> Features are decomposed into stories All stories are estimated All stories have a clear definition of done 	<ul style="list-style-type: none"> All stories follow the INVEST principle All stories are prioritised and can be linked to feature 	<ul style="list-style-type: none"> All stakeholders (Analyst, Developer, Tester, Subject Matter Expert) are involved in defining and elaborating stories Testable acceptance criteria are agreed for each story 	<ul style="list-style-type: none"> Customers are involved in defining the acceptance criteria All story risks are identified and addressed through technical 'spikes' 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community 		
Development	Iterative Development	<ul style="list-style-type: none"> Work is NOT broken small units that can be completed within an iteration Iterations are either not used or work is not reviewed and planned on a regular basis 	<ul style="list-style-type: none"> Work is broken in fixed-time iterations Work/stories are completed within an iteration 	<ul style="list-style-type: none"> Each iteration delivers working software that can be demonstrated to the customer Work not completed within an iteration does not contribute to measured progress 	<ul style="list-style-type: none"> Each iteration delivering working software that is acceptance tested by users 	<ul style="list-style-type: none"> Each iteration delivers working software that could be confidently deployed to production environment 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community 		
	Code Management	<ul style="list-style-type: none"> Source code is NOT version controlled Test and development artefacts are stored on a local file system and are not widely accessible to all team members 	<ul style="list-style-type: none"> Source code is version controlled 	<ul style="list-style-type: none"> All changes are linked to an issue / story / defect tracking system Test artefacts are stored electronically and are accessible to all team members 	<ul style="list-style-type: none"> Changes are committed to the version control system at least once per day All test code / scripts / cases are stored together with the relevant production code in the version control system 	<ul style="list-style-type: none"> Small changes are committed multiple times per day Tagging, branching and merging is used to isolate changes from main-line Advanced version control techniques are used 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community 		
	Development Practices	<ul style="list-style-type: none"> Coding standards NOT are defined and followed Source code is NOT modular and readable with minimal complexity and duplication Units tests are NOT written to verify basic functionality 	<ul style="list-style-type: none"> Coding standards are defined and followed Source code is modular and readable with minimal complexity and duplication Unit tests are written to verify basic functionality 	<ul style="list-style-type: none"> Significant code changes are peer reviewed Unit tests are automated and follow FIRST principles 	<ul style="list-style-type: none"> Static code analysis (complexity, duplication, query execution plans, ...) are used regularly to assess code quality and identify technical debt 	<ul style="list-style-type: none"> Pair-Programming techniques are commonly used Refactoring is common to ensure technical debt is maintained at an acceptable level 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community 		
	Integration & Deployment	<ul style="list-style-type: none"> Unreliable or inconsistent migration and deployment processes Only select team members can migrate and deploy changes No use or understanding of Continuous Integration or automated build practices 	<ul style="list-style-type: none"> A repeatable and reliable process exists for migrating and deploying code changes All developers are able to migrate and deploy changes to test environments. 	<ul style="list-style-type: none"> A CI system is used build and execute all tests as code is committed to the version control system Build artefacts are packaged and used to deploy to pre-production 	<ul style="list-style-type: none"> The CI system provides continuous feedback on a big visible display Any failures attract immediate attention Build artefacts are packaged and used to deploy to production 	<ul style="list-style-type: none"> The CI system executes non-functional and can deploy changes to test environments Build artefacts are packaged and scripts are used to deploy to production with minimal intervention 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community Production deployments can be completed and rolled-back with a single click 		
TEST	Test Planning	<ul style="list-style-type: none"> No agreed testing approach within the team Tests are NOT considered a valuable asset 	<ul style="list-style-type: none"> There is a defined approach to testing Tests are considered a valuable asset that can be reused as development changes are implemented 	<ul style="list-style-type: none"> Test strategy is agreed and understood by the team and incorporates specific targets (coverage, defects, ...) Test strategy is documented and visible to all team members (BVC) 	<ul style="list-style-type: none"> Non-functional requirements (performance, reliability, ...) are agreed with the customer and included the test strategy Test planning is based on risk associated with technical complexity and business impact Back out/Roll-back plans exist and are verified 	<ul style="list-style-type: none"> Test artefacts are treated with the same importance as code and continually refactored and maintained The relevant _____ility (usability, maintainability, supportability ...) and security tests form part of test strategy 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community 		
	Test Execution	<ul style="list-style-type: none"> Tests are NOT executed on a regular basis State of current defects is unknown 	<ul style="list-style-type: none"> Functional testing occurs regularly All testers can execute the functional tests Production verification testing is used to ensure success deployments 	<ul style="list-style-type: none"> A majority of functional tests have been automated Exploratory testing forms part of test execution Customers verify implemented features prior to deployment 	<ul style="list-style-type: none"> A majority of non-functional tests (performance, reliability, ...) are completed prior to deployment Customers verify implemented features as they are completed 	<ul style="list-style-type: none"> Functional and non-function testing occurs continuously within development iterations 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community 		
	Defect Management	<ul style="list-style-type: none"> Defects are NOT recorded Total number of defects is unknown 	<ul style="list-style-type: none"> Defects being raised and tracked as part of test execution processes 	<ul style="list-style-type: none"> Defects associated with features/changes in development are identified and resolved within the same iteration 	<ul style="list-style-type: none"> Defect remediation activities are incorporated into iteration planning Story (progress) points are NOT claimed for defect remediation activities arising from developed features 	<ul style="list-style-type: none"> Defect trends are highlighted based on metrics and used to drive continuous improvement initiatives across iterations 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community 		
	Test Environments	<ul style="list-style-type: none"> Environments are NOT available to properly execute tests 	<ul style="list-style-type: none"> Test environments are available that isolate testing from development changes Test environments are available for sufficient time to support regular testing 	<ul style="list-style-type: none"> Functional testing occurring in an environment that is regularly updated with development changes Testers have access to same toolset as developers with appropriate hardware to support 	<ul style="list-style-type: none"> Functional testing occurs in an environment that is updated as features are completed Test environments isolate external system interfaces while still mimicking behaviour 	<ul style="list-style-type: none"> Test execution includes a pre-production environment which is architecturally equivalent to production Test environments scale to support growing number of tests without increasing execution time 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community 		
	Test Reporting	<ul style="list-style-type: none"> There is no visibility of system health amongst team 	<ul style="list-style-type: none"> Test results are published and visible to all team members and stakeholders Test results are weighted by defect severity 	<ul style="list-style-type: none"> Test results/metrics are being used to guide planning activities Big Visual Displays being used to demonstrate system health NOT just defect numbers 	<ul style="list-style-type: none"> Historical test metrics/reports are stored and readily available Test metrics are used for trend analysis that helps the team continuously improve productivity and quality 	<ul style="list-style-type: none"> Test metrics/report data being used to identify quality trends provide all team members and stakeholders with a clear view of system health and confidence in production readiness 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community 		
	Operational Support	<ul style="list-style-type: none"> Production usage of the system is NOT monitored No SLAs (informal or otherwise) exist for the system or service Frequent support incidents or events occur 	<ul style="list-style-type: none"> System performance is monitored for incidents and problems Process exists for prioritising and resolving issues 	<ul style="list-style-type: none"> Automated monitoring of system outages and events SLAs exist between to customer and support team Automated notification for incidents and problems Affected users/stakeholders are properly communicated on of issue progress and resolution 	<ul style="list-style-type: none"> Performance metrics for system or service are captured and reported Customer satisfaction is regularly captured and reported Remote access exists to support production system/service 	<ul style="list-style-type: none"> Performance metrics and customer satisfaction are used as the basis for continuous improvement SLAs contain targets that are compared and reported against actual measurements 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community 		
TEAM	Risk & Issue Management	<ul style="list-style-type: none"> There is NO awareness of current risks or issues 	<ul style="list-style-type: none"> Current risks are identified Current issues are raised by the team and classified according to impact Blocking issues are addressed as a priority by Team Leader/Manager 	<ul style="list-style-type: none"> Risks are raised as they are identified The team aware of all risks and issues and who has responsibility for resolution 	<ul style="list-style-type: none"> Risks and issues are raised as they are identified and are managed in structured approach The customer is aware of all risks and issues 	<ul style="list-style-type: none"> The financial impact of risk and issues understood The customer participates in developing mitigation strategies 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community 		
	Technical Debt	<ul style="list-style-type: none"> Features take longer and longer to implement Tests consistently fail System development is unsustainable 	<ul style="list-style-type: none"> The concept of technical debt understood by all team members 	<ul style="list-style-type: none"> Technical debt issues are manually identified and fixed on as-needs ad-hoc basis 	<ul style="list-style-type: none"> Decisions about work priority are guided by technical debt issues Effort is often reserved purely to pay-down technical debt 	<ul style="list-style-type: none"> Technical debt is tracked using automated measures and continuously addressed as part of each iteration 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community 		
	Team Roles	<ul style="list-style-type: none"> Team members work ONLY within defined roles Balance of team skills is inappropriate 	<ul style="list-style-type: none"> The team is composed of staff that represent all of the required roles and skills The appropriate proportion of skills available within the team 	<ul style="list-style-type: none"> Analysts, Developers and Testers work together to as a cross functional team and support 	<ul style="list-style-type: none"> The team is able to recruit new members easily The team is involved in the recruitment and selection process 	<ul style="list-style-type: none"> The team is able to easily transition from current work to work of a different type 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community 		
Team Values	<ul style="list-style-type: none"> Agile values are NOT understood or respected Team is ambivalent towards continuous improvement and self-development 	<ul style="list-style-type: none"> All team members understand and exhibit Agile values Retrospectives actions are used to improve team processes/practices for each iteration 	<ul style="list-style-type: none"> Team values are agreed and represented by visible social contract and non-compliant behaviours are called out by team members 	<ul style="list-style-type: none"> The whole team recognise quality issues and prioritises remediation activities 	<ul style="list-style-type: none"> The whole team recognise the importance of self-organization and collaboration 	<ul style="list-style-type: none"> Practices and learnings are shared with the wider community 			