TEST AUTOMATION MATURITY MODEL

T · · Systems ·

TAMM: OVERVIEW

	MANUAL	BEGINNING	DEFINED	ADVANCED	MATURED
Test Development	▪ No test automation ▪ 100% manual test cases	▪ Test cases automated through scripting methodologies ▪ Test automation suite exists, but automatic test cases are not excluded from manual test plan	▪ Test cases developed in Test Framework, which gather test results & logs, generates test reports ▪ Sources of Test Framework and test cases are under version control system ▪ Automatic test cases are permanently used and produce profit value and no crossing with manual test cases	▪ Test cases are classified by levels (unit, integration, system) and types (smoke, sanity, regression, acceptance) ▪ Test Cases are organized in Suites/Test Plans. Test cases and Plans are mapped with requirements. ▪ Test coverage is measurable by test cases and the general level of test coverage by automatic test cases is higher than coverage by manual tests	▪ Script less Test Framework with extremely intuitive GIU can scale across complete QA and Dev teams, test automation tools, development technologies and any complex scenarios ▪ Test Framework implementation re-use some test concept (ex, keyword-driven, data-driven etc.). Test cases can be developed by a manual QA Engineer and fits in test automation pyramid (UI-Middle-Unit level) ▪ 100% of Test cases are automatic
Test Strategy	▪ No formal documentation and processes, odd documents	▪ Defined Release planning and list of preparations processes	▪ Test Automation strategy, goals and scope. Permanent test automation process analysis and improvement	▪ Test coverage and Requirements traceability matrix. Existed workflow of test coverage assessment	Test automation ROI analysis, processes analysis and KPIs: ▪ automation execution time per code commit ▪ quarantine tests that have random success ▪ PASS-FAIL rates per code commit ▪ capture test case duration and CI duration time
Test Execution	▪ Manual or no Test Results ▪ Logs are not collected ▪ Manual test reports	▪ Test Results and Logs are collected and saved locally or send via attachment in test report ▪ Manual test report	▪ Results and Logs collected and automatically saved/uploaded in common test server ▪ Reports are automatically prepared and sent	▪ Existence of User Interface, where we can see test cases and whole test execution history, get any statistics (% passed, failed etc.) ▪ Collect metrics ▪ Possibility of search by entries/string in test logs	▪ Automatic analysis of Test Results and Logs, categorization of errors and prediction from regressions/issues
Defect Managing	▪ No bug and task tracking systems are used	▪ Defined defect tracking process with manual bugs and tasks creation	▪ Full integration with bug tracking and test management systems ▪ Defined bug triage process.	▪ Defects are trapped and sent directly back through the fix, test creation, and regression test processes ▪ QA Test team is an integral part of product development (QA and Dev work together)	▪ Automatic defect prevention system (which is explore and directly reverts regressions) ▪ Risk-based testing (RBT) system is used
DevOps Integration	▪ Manual delivery of software changes once in 4 months. ▪ Manual installations of application ▪ One monolithic application ▪ No build versioning	▪ Semi automated delivery of software changes once in 4 months ▪ Semi automated installations. Versioning of the delivered builds	▪ Semi automated deliveries of versioned software on demand ▪ Automated installations with rolling updated ▪ Automated version tagging of the builds	▪ Automated continuous delivery of software after implementation and test ▪ Automated installations without downtime ▪ Feature branch development with auto merging ▪ Splitting the deliveries into stateless services (components).	▪ Automated continuous delivery of software after implementation and automated test ▪ Automated rolling installation without downtime of new versions of components. ▪ Backwards compatibility of the versions between components ▪ Automated scaling of services ▪ Master branch development with feature toggling

T · · Systems ·

TAMM: TEST DEVELOPMENT

Manual	Beginning	Defined	Advanced	Automated / Matured
No test automation 100% manual test cases	Basic level of automation Test cases automated through scripting methodologies Test automation suite exists, but automatic test cases are not excluded from manual test plan	Test cases developed in Test Framework, re-use common classes or libraries Test Framework gather test results and logs, generates test reports Sources of Test Framework and test cases are under version control system Automatic test cases are permanently used and produce profit (value), have no crossing with manual test cases Test data management (automatic data generation)	Test cases are classified by levels (unit, integration, system) and types (smoke, sanity, regression, acceptance) Test cases are organized in well defined test plans, suites or scenarios Test cases and Test plans are mapped with actual requirements. Test coverage is measurable by test cases The general level of test coverage by automatic test cases is more/higher than coverage by manual test cases	Script less Test Framework with extremely intuitive GIU can scale across complete QA and Dev teams, test automation tools, development technologies and any complex scenarios Test Framework implementation re-use some test concept (ex, keyword-driven, data-driven etc.). Test cases can be developed by a manual QA Engineer and fits in test automation pyramid (UI-Middle-Unit level) 100% of Test cases are automatic

T · · Systems ·

TAMM: TEST STRATEGY

Manual	Beginning	Defined	Advanced	Matured
Some odd documents, no formal documentation and processes	Defined (& maintained) Release planning and list of preparations processes	Defined Test Automation strategy, goals and scope/user stories Permanent test automation process analysis and improvement	Detailed Test coverage and Requirements traceability matrix. Established workflow of test coverage assessment	Test automation ROI analysis, processes analysis and KPIs: - automation execution time per code commit - quarantine tests that have random success - PASS-FAIL rates per code commit - CI duration time - capture test case duration (if it exceeds some max time)

T · · Systems ·

TAMM: TEST EXECUTION

Manual	Beginning	Defined	Advanced	Matured
Test results somewhere saved Logs are not collected, No test reports or manual prepared	Results and Logs collected and saved locally or send via attachment in test report Manual test report	Results and Logs collected and automatically saved/uploaded in common test server Reports are automatically prepared and sent	Existence of User Interface, where we can see test cases and whole test execution history, get any statistics (% passed, failed etc.) Collect metrics Possibility of search by entries/string in test logs	Automatic analysis of Test Results and Logs, categorization of errors and prediction from regressions/issues

T · · Systems ·

TAMM: DEFECT MANAGMENT

Manual	Beginning	Defined	Advanced	Matured
No bug and task tracking systems are used	Defined defect tracking process with manual bugs and tasks creation	Full integration with bug tracking and test management systems Defined bug triage process	Defects are trapped and sent directly back through the fix, test creation, and regression test processes QA Test team is an integral part of product development (QA and Dev work together)	Automatic defect prevention system (which is explore and directly reverts regressions) Risk-based testing (RBT) system is used

T · · Systems ·

TAMM: DEVOPS INTEGRATION

Manual	Beginning	Defined	Advanced	Matured
Manual delivery of software changes once in 4 months. Manual installations of application One monolithic application No build versioning	Manual or semi automated delivery of software changes once in 4 months. Semi automated installations. Versioning of the delivered builds	Semi automated deliveries of versioned software on demand Automated installations with rolling updated Automated version tagging of the builds	Automated continuous delivery of software after implementation and test Automated installations without downtime Feature branch development with auto merging Splitting the deliveries into stateless services (components)	Automated continuous delivery of software after implementation and automated test Automated rolling installation without downtime of new versions of components. Backwards compatibility of the versions between components Automated scaling of services Master branch development with feature toggling

T · · Systems ·

TAMM: THEORY

Test Automation Maturity Model (TAMM) re-uses approach of some well known models and concepts, like Capability Maturity Model (CMM), Capability Maturity Model Integration (CMMI) and Testing Maturity Model (TMM).

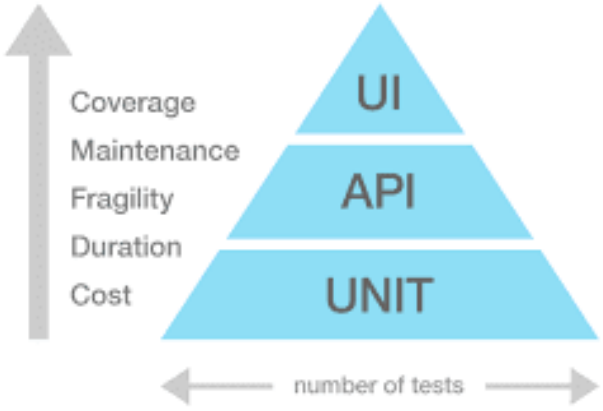
TAMM is aimed to provide a framework for evaluation the maturity of the test automation processes in an organization or projects, classification of test automation level and providing targets on improving maturity.

TAMM evaluation includes evaluation by five technical criteria's and results guidelines (advisability) based on return of investment, project needs and technical aspects. TAMM does not evaluate Level of Goodness or Badness of Automation.

T · · Systems ·

AUTOMATION COMPETENCE CENTER GUIDELINES

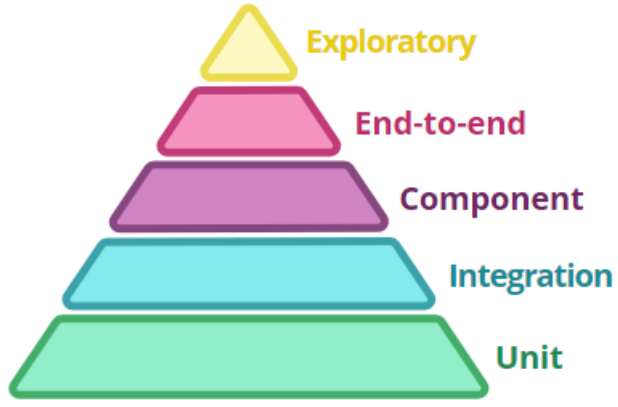
Classic Functional Test Pyramid



UI Level (FrontEnd): Selenium, Python mechanize, UFT (Unified Functional Test)

Middle Level (BackEnd): Cucumber, SOAP UI, Django, Xyna Test Factory

Unit Level (Developers): xUnit family frameworks, like JUnit, PyUnit, PHPUnit



Non Functional test:

Load / Performance: JMeter, XYNA Test Factory,

Stability / Stress: JMeter

Security: Nmap, Nessus,

T · · Systems ·