telbit

testudio

testudio

TeStudio is a system whose main purpose is to lighten the tasks related to software quality assurance (SQA) in the software development activity.

TeStudio designers are thrived in continuously developing a system that follows and gathers the best practices being adopted worldwide, related to software quality assurance activities, without forcing the adoption of any SDLC methodology or pre-built process.

TeStudio creates key information flow paths, which enables symbiotic relations between stakeholders, developers, testers and SQA roles.

Currently, **TeStudio** is composed by four main modules each one with a well defined purpose and focused on a software quality assurance activity:

testudio **requirement**manager

testudio **automation**lab

testudio **defect**manager

testudio **data**analytics

TeStudio's mission is to provide software development groups the ability of delivering quality software products without the hassle of making them aware of a painful and heavy SQA process.

TeStudio is focused on continuous Quality Improvement by providing means to evolve the following concurrent processes following the Deming Wheel (Plan-Do-Check-Act) approach:

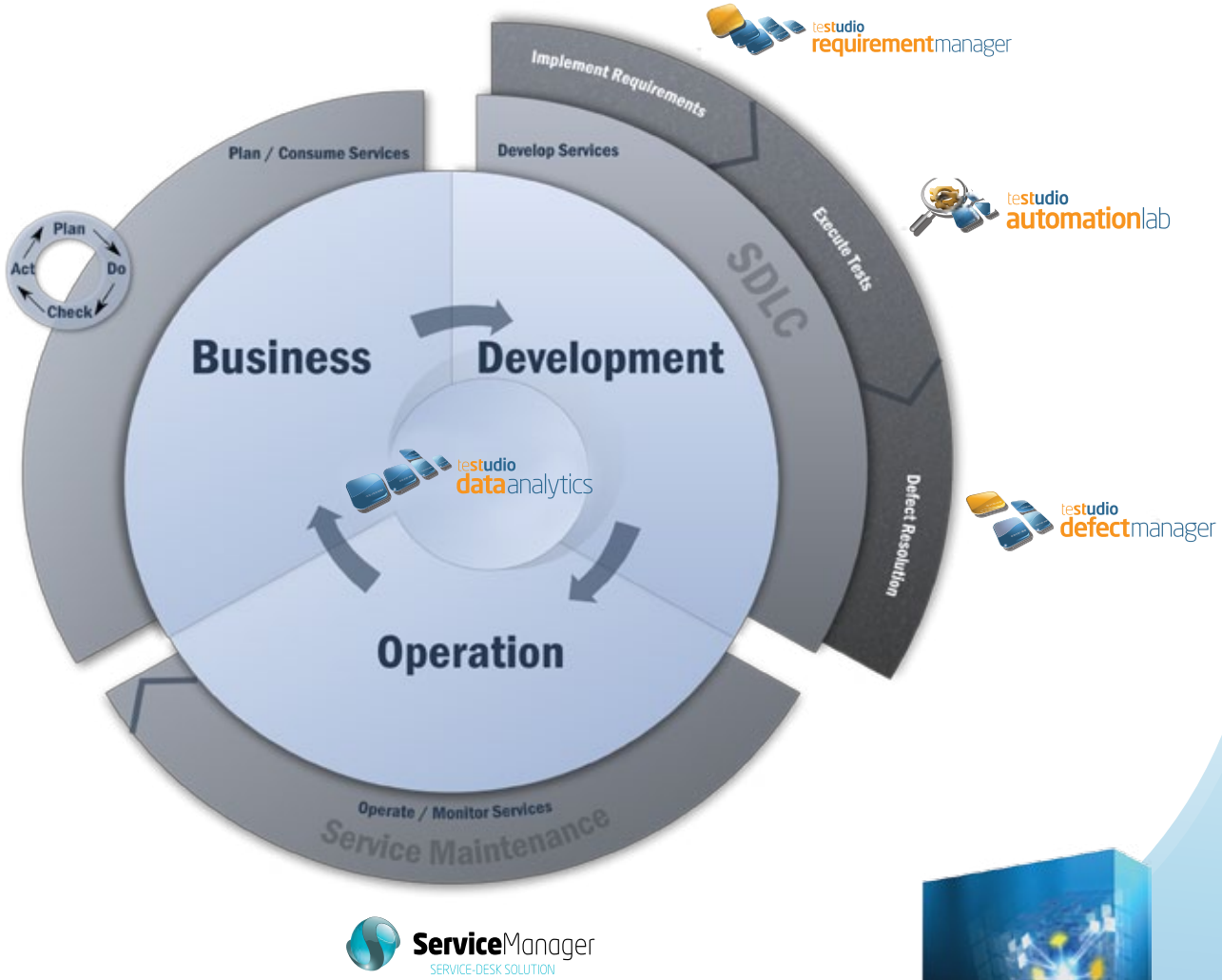
- processes of software development lifecycle activities
- processes of defect detection, removal and prevention

Although **TeStudio** is a family of products, it is not necessary to adopt all the four modules to build a Software Quality Assurance solution.

Each **TeStudio** module handles a number of 3rd party integrations that unlocks the potential of implementing custom SQA systems.

These integration capabilities are also used to plan phased integrations of **TeStudio**.

TeStudio provides an out-of-the-box integration with Service Manager which is Telbit's service management web solution for small to medium sized businesses and organizations. **Service Manager** is fully oriented to deal with support activities related to software development industry.



testudio requirementmanager

Requirements are definitions of the boundaries, constraints and needs of a software product. One can see the requirements as the clauses of a formal contract between the development team and the interested part in a software product.

TeStudio Requirement Manager (**TSRM**) is in charge of easing the activities regarding the systematic handling of requirements through a dedicated interface built for this purpose.

By integrating **TSRM** with the other **TeStudio** family products, requirements gain pivoting capabilities, and the user can reach the defects and tests related with the focused requirement.

REQUIREMENTS SOURCES AND ELICITATION

Since the very first contact with a client, the development team starts to acquire useful information vital to the success of the upcoming product. All the information gathered that is relevant to the software project must be registered. The collected information is part of requirements elicitation activity, and it's relevant to the requirements analysis and specification.

TSRM encourages a systematic handling of the information that flows between the client and the development team, without super-imposing any type of methods to the gathering of information.

The methods can go from interviews, prototypes, meetings to observation, and the sources from domain knowledge, stakeholders to organizational environment that supports a business process.

REQUIREMENTS ANALYSIS AND SPECIFICATION

TSRM provides the means to make an easy management of requirements, since it creates a mediation point in the communication flow between the project stakeholders and the development team.

Requirement classification, conflict resolution and monitoring, dependency and hierarchy analysis, prioritization, requirements packaging are some of the concerns of **TeStudio** Requirements module.

The knowledge regarding a project is built upon each interaction that is made with the requirements database. **TSRM** covers several problem areas regarding requirement attributes such as Discrimination (status and priority), Quality (unambiguous, complete, verifiable, consistent and traceable) and Risk (feasible and dependable).

CHANGE MANAGEMENT, TRACING AND MONITORING COVERAGE

Software Quality Assurance starts on proper and effective Requirements Management. With **TSRM**, every interaction with a requirement that results in some kind of change is recorded with the necessary detail that permits a detailed and accurate configuration management of all SQA related activities.

Auditing and tracing activities can be done independent of the particular interveners related to the item under attention.

REQUIREMENT METRICS

Requirements coverage monitoring as well as other key indicators are essential to keep control of the defined SQA levels as well as project evolution regarding SQA related indicators - test coverage, test count, defect removal rate, defects main causes, requirement risk management, etc.

TeStudio provides a data analytics module, which has a single purpose: to concentrate all the responses to the answers regarding Software Quality Assurance Metrics.

By concentrating the right amount of information in one single place, **TeStudio Data Analytics** brings visibility of the project key indicators to all the people who matters.

REQUIREMENTS VALIDATION AND VERIFICATION

The validation of the system being developed is a continuous process that requires the stakeholders interaction with the system being developed. The stakeholders of a project should validate if the system goes towards their needs. **TSRM** is a privileged contact point between the stakeholder and the system being developed.

As stated before, a requirement defines a boundary, constraint or need, thus requirements are the verification points of a product in development. The product being delivered against the specified contract should be verified regarding each requirement. This is where Requirements Management and Software Testing disciplines intercept. One must conduct an appropriate verification of every requirement, preceding the delivery of a product.

These verifications are conducted in **TeStudio Automation Lab**.

testudio automationlab

Since testing is one of the critical stages on the Software Development Lifecycle, **TeStudio** has a module specially purposed to design, manage, run and evaluate software functional tests - **TeStudio Automation Lab** (**TSAL**).

Our vision to this module is - from the very start - to build a system which fosters testware knowledge spreading among a software testing team and the remaining stakeholders like developers and project managers. Besides the information system component, this module also allows testers to specify, run and evaluate software tests.

Automation Lab has built in modules suitable to test Web, Microsoft© .NET Framework stand alone applications and also applications which run on a Java Runtime Environment™.

These tests lie on fully automating the target application's GUI emulating a user/tester behavior.

Automation Lab also provides support to design and execute tests targeting several RDBMS. Despite this, **Automation Lab** is not tied up to any specific SQL dialect, enabling users to use whichever one they find most appropriate to each particular scenario.

Moreover, **Automation Lab**'s architecture also allows testers to design and execute tests targeting local or remote command line applications and also web services (e.g. SOAP or REST) given that there is available a "proxy" process ready to invoke their published operations and output their results.

ANATOMY OF A TEST

Tests on **Automation Lab** can just be a textual script to guide the tester. On the other hand, they can be a set of fully automated scripts which are automatically executed and evaluated without testers' intervention. These approaches are by no means mutually exclusive since **Automation Lab** also supports hybrid tests i.e., tests composed by manual and automated steps. It's up to the test designer to judge the best tradeoff regarding manual versus automated test execution and/or evaluation.

Tests are designed combining a set of atomic and reusable steps in a suitable workflow.

For example, in a scenario of testing one given requirement in a classical three tier application, composed by a RDBMS backend, an application server and a GUI front end, a test is likely to include steps performing the following tasks: put the backend database in a known state, capture application server logs as long as the test is running, exercise some system's GUI functionalities and finally, assert the actions' side effects on the backend database. In a fully automated scenario, each of these steps may be independently evaluated against previously defined pass and/or failure conditions. Each step's test oracle is expressed using regular expressions.

As you can guess from the above example, one single test may be composed by several kinds of steps: SQL, Bash, GUI automation scripts, etc. Each of those types of steps has its own way to express the procedure to be executed on its target. All but the GUI automation scripts are already known by the average test designer since they are common and standard ways to interact with systems under test.

GUI automation scripts are specified in a simple yet powerful Domain Specific Language specially designed to express actions on GUIs' widgets.

Most of these scripts may be created using the capture/playback features provided by **Automation Lab**.

ENVIRONMENT DECOUPLING

Since test steps will be executed on databases, application servers, remote hosts, local applications, etc, **Automation Lab** provides a way to store and manage this information. The set of target hosts characterization is called an Environment and each step has a weak reference to their target host using a named alias.

This way, it is possible to setup several test Environments, and right before start executing the tests, the tester can pick up one of the several Environments available to execute the test set.

A side effect of this feature is that this way, **Automation Lab** ends up to store in a centralized way the characterization of the entire test Environments in a testing department.

AUTOMATE STEPS' EVALUATION

Heavily conditioned by the nature of the system under test and also by the functional requirements being tested, there could be more or less tests/steps suitable to be automated. **Automation Lab** allowing manual, automatic and also a hybrid approach to test execution relies on the tester to judge the best tradeoff in each particular scenario. However, **Automation Lab** has a unified approach to define the test oracles which already proved to be an effective way of automating steps evaluation. All types of steps, manage to output text from their execution. Based on this principle it is natural to use regular expressions to characterize the expected steps' outcome.

These can be as simple as a string to match an exception or error codes on a trace log or otherwise, complex enough to perform an elaborate pattern matching to express more demanding evaluation scenarios.

Although steps automation has an obvious upfront cost, when properly done - in most of the cases - it increases the tests' resilience to the human ad-hoc evaluation shortcomings in a dramatic way.

REGRESSION TESTING

Experience shows that along the development and maintenance stages of a software project, hardly any test project is a complete fresh start.

Automation Lab allows previously created testware regarding similar/ previous projects to be (re)used to make the base of newer runs.

This is a critical issue when performing regression testing, due to the high degree of overlap on the amount of tests to be re-executed between two versions of the same system.

Automation Lab really encourages a test set to be built based on previous ones (eventually updating or even removing some legacy items). Since full testing a reasonably complex system is a potentially endless task, this continuous improvement, turns the test set more effective along the time and better suited to find (at least) the most harming defects. This also frees the testers to improve the tests quality instead of being worried "reinventing the wheel" every time a new test cycle begins.

TEST RUN EVIDENCES

Automation Lab provides the development team properly formatted test reports which contain all tests' Environment characterization (target hosts including user credentials, etc) pre/post conditions, executable content and eventual traces/log files content. These collected evidences are an essential asset to attach to defect records easing the developers' diagnostic job and therefore turn the defect correction more expedite. In the case of tests which run as expected these evidences are also stored for future audit purposes.

EXTENSIBILITY

Given the diversity of systems under test, their architectural structure and interfaces' heterogeneity, it is not realistic to build a tool aiming, only on its own to be suitable to perform tests on all platforms in the most effective way. To overcome this issue,

Automation Lab provides an extensibility mechanism which allows software vendors to create and integrate custom utilities to be executed and evaluated by its test execution/evaluation engine.

testudio defectmanager

Software Quality Assurance is defined as the systematic approach to the evaluation of the quality and adherence to agreed upon standards, processes and procedures. The flaws identified and registered counts as defects of a product, and its quality is usually measured by the number of its defects. The anomalies of a software product should be carefully controlled since it can seriously affect the user experience of the developed product, as well as the image of the responsible developing team/ company.

TeStudio Defect Manager (**TSDM**) manages a database of defects found since the very start of the development phase as well as their lifecycle.

DEFECTS IDENTIFICATION AND CHARACTERIZATION

Once a defect is identified **TSDM** provides a registration point for all the details related to the identified defect. The defect identification is accomplished by determining a set of key properties along the defect lifecycle: description, severity, priority, detected by, assigned to, cause, origin, state, etc.

An adequate characterization of defects leads to a better understanding of the product, facilitates its correction, and creates information suitable for being consumed by project intervenient (development responsible or customer).

Rather than build a complex system that predicts all the possible cases of characterization of a defect, **TSDM** offers customization mechanisms that allows an adaptation to different development methodologies or processes. **TSDM** can be adjusted to the process defined to deal with defects, and not the other way around. One can customize the defect workflow definition, and the number and type of fields that characterize a defect.

QUALITY IMPROVEMENT INDICATORS

As well as with Requirements and Tests, **TeStudio Data Analytics** brings visibility to defect related metrics such as creation rate, correction rate, defect state, main defect causes, requirements having defects, etc. By having access to key metrics it is possible to plan and design preventive and corrective measures as well as checking its effectiveness.

By iterating along the Plan-Do-Check-Act wheel, it's possible to evolve and improve the business processes defined in an organization.

UP-TO-DATE PICTURE

Defects can be monitored and tracked by taking in account the large number metrics abstracted from the project details, but it also provides the mechanisms to keep the defects intervenient - usually developers and testers - with up-to-date information of a defect evolution.

The involvement, commitment and motivation levels of a team are benefited if they have a clear and broad picture of the project and how the individual work contribute to push the project forward.

RSS subscription of a defect or a set of defects and e-mail notifications are a few examples that can be mentioned regarding defects tracking and monitor mechanisms.

TRACING BACK TO REQUIREMENTS, TESTS AND DEFECT CAUSES

TeStudio has been built upon the principle that the information regarding SQA and software development activities must always be easily accessible to the people who matters.

The cohesive integration of all the **TeStudio** modules: **TeStudio Requirement Manager**, **TeStudio Automation Lab**, **TeStudio Defect Manager** and **TeStudio Data Analytics**, allows that an intervener with each **TeStudio** module easily reaches information that is handled in another module.

As an example of this cohesive integration take for example the possibility to see which test was in the origin of a defect and which requirements it affects.

testudio dataanalytics

TeStudio Data Analytics (**TSDA**) is a web analytics solution that brings visibility to Software Quality Assurance metrics.

It enables the analysis of management actions' effectiveness and enriches the development teams with project evolution awareness capabilities.

Among other characteristics, one can highlight:

DATA VISIBILITY

- Bundled Management KPI's regarding Software Quality Assurance
- Data accessibility through the web-browser and information push through connection tools (RSS, Email)
- Embedded export mechanisms

CUSTOMIZATION NEEDS

- Extendable mediation database to support external sources of information
- Customization mechanisms the allows the adaptation of the data analytics to the company needs

3RD PARTY INTEGRATIONS

- Open API that permits data integration with other systems

testudio requirementmanager

testudio automationlab

testudio defectmanager

testudio dataanalytics

testudio

Your Software Quality Assurance Solution

telbit

www.telbit.pt