LogiGear

Leverage Test Automation effectively with TestArchitect

LogiGear

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This Implementation Guide outlines major steps to empower testers with diverse skill sets and succeed at test automation with TestArchitect™. With some tailoring, you can turn your investment into tangible profits as quickly as possible while most importantly, causing minimal disruptions.

The Preparation Stage describes related parties you should involve, how to deploy TestArchitect component by component, and the available license schemes of which you can make use.

The Execution Stage shows you the possible roles on a team, which automation skills your team should master, test activities and test assets you will manage. The rest of this guide lists the best practices and other useful resources and services which you can leverage to succeed with TestArchitect.

PREPARATION STAGE

Involving These Parties

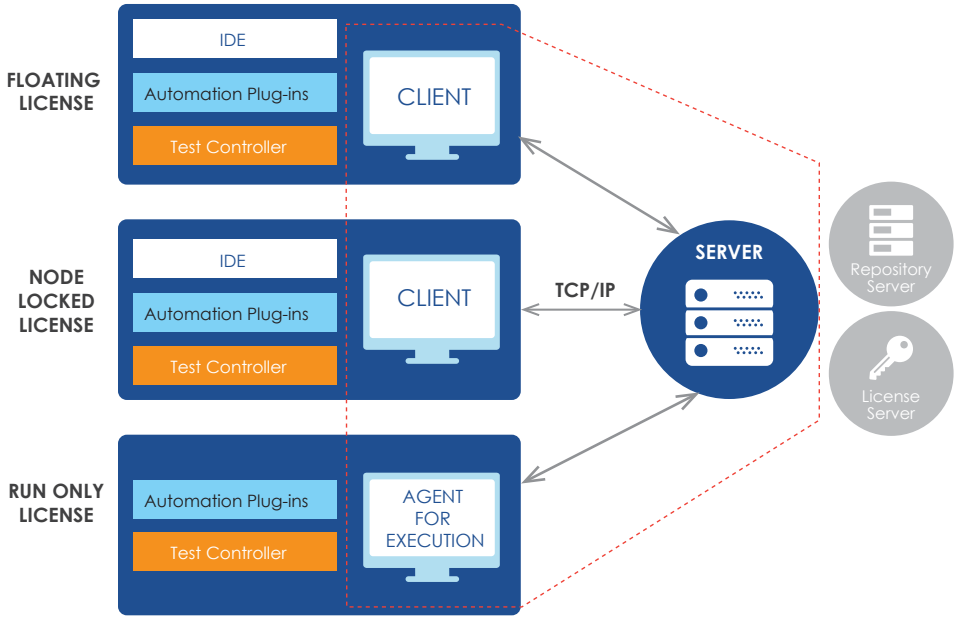
The table below describes the parties who will most likely take part in this initiative.

Role	Involvement
Testers & Automation Engineers	Responsible for learning new automation skills, migrating existing test artifacts, building new test assets with the new methodology, and ultimately, making the most of the new framework
Developers	Responsible for ensuring testability of the application under test (AUT), helping testers in deploying a complete Continuous Testing process, and extending the automation libraries if need be.
Operations (IT)	Responsible for setting up the network infrastructure, assigning domain names, managing and maintaining important components, such as Repository Server, License Server and back-up servers.
Product Management	Responsible for adjusting the team's backlog to accommodate the transformation since it almost certainly affects the delivery capability of the team as a whole.
Line Managers	Responsible for introducing new work processes and roles, reallocating resources, coaching existing and new members in their new endeavors, etc.

Deploy TestArchitect

Deployment Model

The following picture illustrates a typical network topology for a small test team.



NOTE: All machines must be in the same network (LAN or VPN)

Install TestArchitect

The Deployment Model consists of the following TestArchitect components.

Component	Description
Repository Server	The main database storing all of your test assets. One server machine should be dedicated to host this component. NOTE: • A back-up Repository Server is also desirable in case the main server stops working. • Repository Server also hosts other TestArchitect components such as Lab Manager and Dashboard
License Server	Responsible for issuing licenses to Client machines and Test Controllers. License Server Control Panel displays license information and status as well as provides features to manage the License Server instance and purchased licenses. NOTE: A back-up License Server is also desirable in case the main server stops working
Client machines	One or more work stations with TestArchitect Client (IDE) installed, on which testers can author and execute tests, review results, generate reports, etc.
Execution machines	Real, virtual or cloud machines with TestArchitect Test Controller and necessary automation libraries installed. By default, Client machines can also execute tests.

NOTE: All of TestArchitect components above are packaged in only one installer for easy installation. For the step-by-step instruction, refer to Installation Guide and License Server Installation Guide

Types of Licenses

To monitor license usage and plan for purchasing new licenses if necessary, it's good to understand the available license schemes in TestArchitect.

Scheme	Meaning
Floating	May be shared among members of an organization and is not tied to any one particular person or machine. At any given point in time, a single floating license can only be used by one host. E.g. if your team has 5 members but only 3 floating licenses, only 3 members can use TestArchitect concurrently.
Node-locked	Is locked to a specific machine permanently. When that machine is not using the node-locked license, the license goes unused. A node-locked license can be transferred from one machine to another, but the frequency of such transfers is limited.
Run-Only	This license scheme is dedicated for execution of tests and reporting of results. You cannot develop tests or manage test assets using the IDE with this license scheme.

EXECUTION STAGE

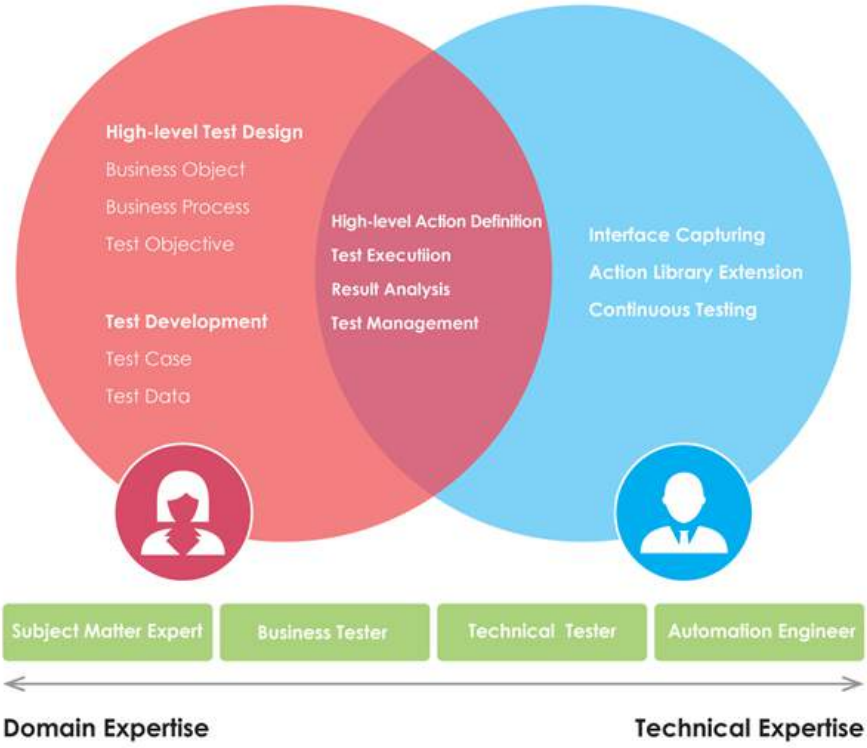
Build Your Team

Assign Roles

In Action Based Testing (ABT), there are several overlapping roles with different skill sets ranging from Domain Expertise to Technical Expertise.

- Subject Matter Expert
- Business Tester
- Technical Tester
- Automation Engineer

The below Venn diagram illustrates the responsibilities of these roles.



These roles can be described briefly as follows.

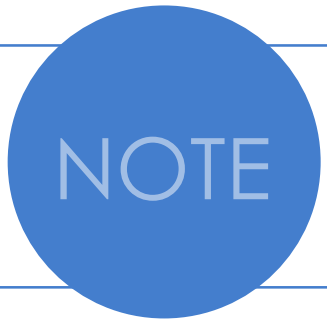
Component	Description
Subject Matter Expert & Business Tester	Responsible for defining Business Objects, Business Processes, Test Objectives, designing Test Modules, Test Data, and working with Automation Engineers to develop High-level Actions, as well as a shared understanding of Business Objects Responsible for designing test cases and validating test results. This role is not mandatory but it's worthwhile to invest in a team of Subject Matter Experts if testing requires intensive domain knowledge (e.g. oil and gas drilling)
Technical Tester & Automation Engineer	Responsible for working with Testers to implement both High-level and Low-level Actions, capturing Interfaces, extending TestArchitect's Action Library, and creating other logistic tools for Continuous Testing if need be
Client machines	One or more work stations with TestArchitect Client (IDE) installed, on which testers can author and execute tests, review results, generate reports, etc.
Test Lead	An Additional role which may arise during the team building process. Responsible for coaching other Testers, providing test plans, module planning, and reviewing the test assets
Automation Lead	Additional role which may arise during the team building process. Responsible for coaching other Automation Engineers, providing automation plans, and reviewing automation assets

Acquire Automation Knowledge

The below table explains the automation skills which each corresponding role should develop to perform their tasks efficiently.

Skill	Description	Role
ABT Fundamental Concepts	Ability to understand ABT's fundamental concepts, such as Test Module, Action, Interface Entity and so on. Since ABT methodology is embodied in TestArchitect, whoever works with TestArchitect test assets should possess this skill. Learn more about ABT here	All
High-level Test Design	Ability to analyze the AUT's business logics to define Business Objects (e.g. "account", "promotion", etc.) and Business Processes (e.g. "rent a car", "check out my cart", etc.), break down test requirements into Test Objectives and assign those Test Objectives to Test Modules in a way that facilitates maintainability and scalability.	Subject Matter Expert & Business Tester
Test Development	Ability to write detailed test steps to achieve the predefined Test Objectives of a certain Test Case. These steps are later translated into Actions (High-level, Built-in Or User-coded Actions). The main goal is to verify various aspects of the AUT with emphasis on the business logics.	Subject Matter Expert & Business Tester

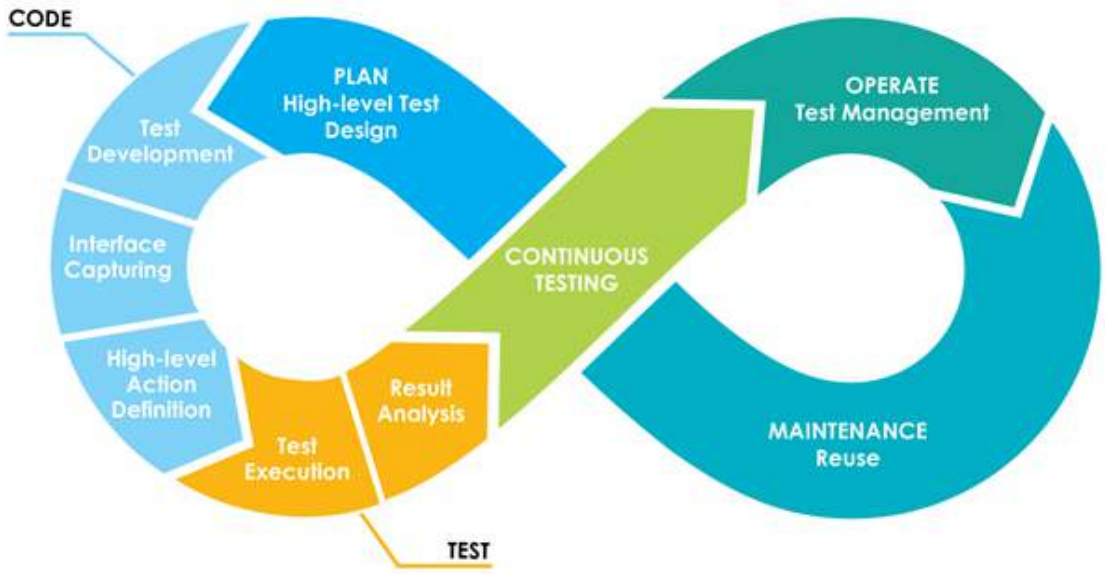
Skill	Description	Role
Test Data Management	A tester must design quality Test Data to manipulate the desired test condition of the AUT and assert its corresponding state. In TestArchitect, Test Data is stored in Datasets.	Subject Matter Expert & Business Tester
High-level Action Definition	Ability to craft reusable High-level Actions based on pre-defined Low-level Actions (Built-in or User-coded actions). A High-level Action can be either an operation (e.g. "log in", "rent car", etc.) or verification point (e.g. "check balance", "check available cars", etc.)	All
Interface Capturing	Ability to understand the AUT's GUI hierarchy to choose non-volatile properties which can uniquely identify the GUI controls of interest. Without Interface capturing, no automation tools including TestArchitect are able to find the controls the test wants to interact with. Mastering Interface Viewer is crucial in this skill.	Technical Tester & Automation Engineer
Action Library Extension	Ability to extend TestArchitect's Built-in Action Library by implementing new User-coded Actions. In addition to basic coding, this skill also consists of insights into how TestArchitect works to make full use of its components, such as Engine API, Automation API, etc.	Technical Tester & Automation Engineer
Test Execution & Result Analysis	Ability to run the tests on demand and analyze the run results. Everyone on the team should possess this skill. TestArchitect already provides an out-of-box toolset for productive result analysis: Screenshot Recorder, Debugger, Result Comparison	All
Continuous Testing	Ability to integrate TestArchitect with other ALM tools if any, e.g. Team Foundation Server, HP ALM, Zephyr, etc., set up automated nightly runs and the execution environments (real machines, VMs, cloud instances, or mobile devices), build logistic tools for Continuous Testing, etc. This skill requires basic coding and familiarity with TestArchitect components such as TAUtilities. At least one member on the team should master this skill. That best candidate would be an Automation Engineer.	Technical Tester & Automation Engineer
Test Management	Ability to understand TestArchitect's test management tools, such as Dashboard, Lab Manager, and Reporting to monitor the project's health in real time	All



- If your team consists of manual testers and they want to transit to a Full-stack Automation Engineer, refer to this article for more details.
- If you need help in building your test team, you can employ LogiGear's Customer Success Programs (learn more).

Deploy TestArchitect

Your team is now ready to apply their newly acquired automation skills and incorporate new ABT test activities into their Software Development Lifecycle. The below infinity diagram describes essential ABT test activities in conjunction with traditional software development phases. With the new accelerated test development speed, your tests can be automated within the same iteration (sprint) as the development team.



Phase	Description
Plan	Product Management (Product Owner), Business Testers and Subject Matter Experts help define the test objectives & test modules based on the team's backlog (list of user stories)
Code	• Business Testers with the help from Subject Matter Experts develop test modules (detailing out test steps to achieve a test condition and assert certain states of the AUT as defined by the test module's objectives) • When more UI designs are finalized and implementations become available, Business Testers add some more test modules to test the application's GUI, navigation and other low-level interactions. • Technical Testers & Automation Engineers capture interface entities • Technical Testers & Automation Engineers with the help from Subject Matter Experts and Business Testers define reusable actions and user-coded actions if need be
Test	• As soon as a test module is fully automated, everyone can run the tests and collect results • Anyone on the team reviews the run results and choose baseline results • Any Tester files bug reports based on the run results. • The team collaborates to gradually increase test and automation coverage to ideally 100% • After developers fix a bug, a tester re-runs the related tests to verify the fix
Continuous	Steps described in the Test phase are repeated automatically

Testing (Build)	Whenever the CI process is triggered, ideally for every code commit. The build's health is monitored through test management tools such as TestArchitect Dashboard, Reporting and Lab Manager. At the end of the development iteration, the team: • Reviews and closes done user stories and tasks • Holds a Retrospective meeting to consolidate lessons learned if need be
Operate	Once the development iteration is done, the application is deployed to production environment and starts its operation. Meanwhile, the team prepares for the next development iteration.At the end of the development iteration, the team: • Reviews and closes done user stories and tasks • Holds a Retrospective meeting to consolidate lessons learned if need be
Maintenance	During the maintenance phase, the team troubleshoots and fixes issues reported from production environment if any. They can also start a new development iteration in which they can: • Reuse existing test assets • Rerun developed tests The team can also spend time on maintaining the test framework: • Constantly refining the test modules to adapt to new changes • Rerun tests on a regular basis to make sure they are "alive". A test not frequently run is a liability, not an asset. • Back up repositories • Monitor resource consumption on server machines to: o Restart them if they are running continuously for too long o Upgrade hardware if need be • Clean up cache • Delete obsolete results on repository or junk local results

Best practices

• Consider your test assets as deliverable products. In Action Based Testing, there are three related but distinguishable product life-cycles, each with its own deliverables:

o system development (main product: software)

o test development (main product: test modules)

o automation development (main product: working and reusable actions)

• Test modules can be developed at any convenient time in the development cycle. Typically it is good to start a test module only when the underlying system knowledge is available. For higher level tests with business functions this can be fairly early, but UI specific tests have to wait until the UI design is in a relatively final and stable stage.

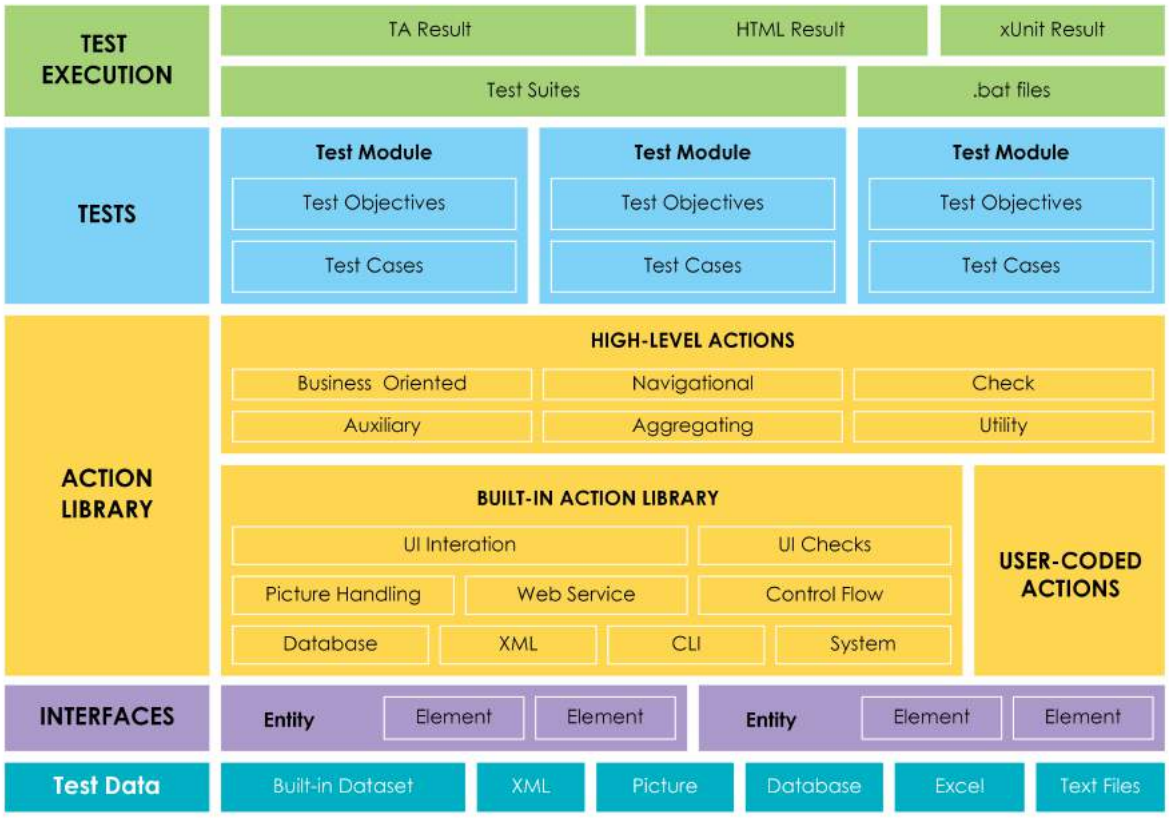
• Modules should only be executed when the system under test is "ready" for them. For higher level modules like functional tests, this means that the lower level tests should have passed, in other words the UI should be stable.

• Aim for the highest degree of test and automation coverage

• Aim for a close cooperation between the members of the team. There should be agreement on what approach to use

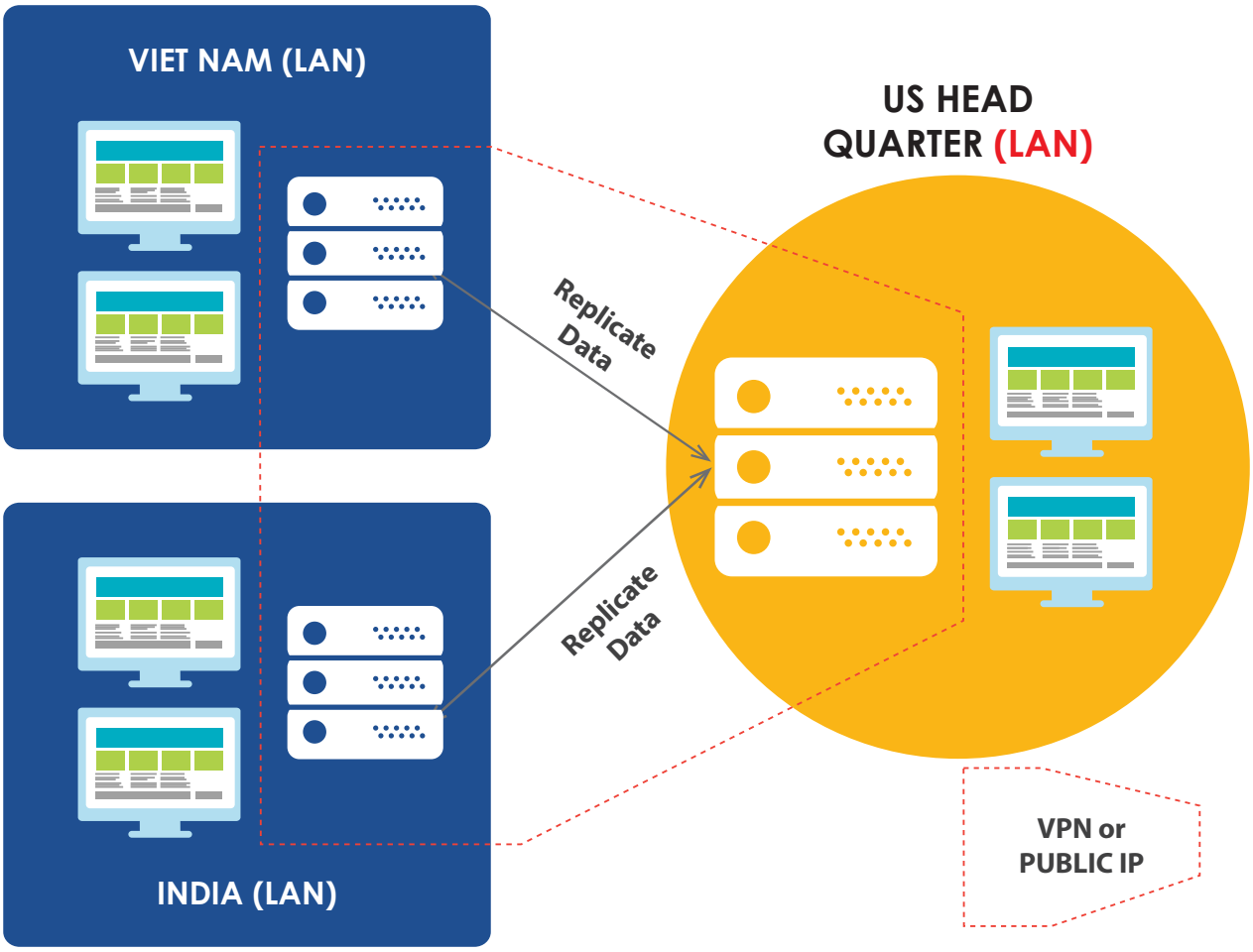
Organize Test Assets

TestArchitect test assets can be categorized as follows:



It's normal for big and long-term test projects to set up several TestArchitect repositories, TestArchitect projects and subscribe one project to another. The table below lists out the basic concerns which need to be taken into consideration when scaling your test project.

Concern	Solution
Co-location	If the team is dispersed across multiple geo-locations, you'll need to design a sustainable physical and logical infrastructure. For instance, from each geo-location, it's recommended to set up one physical server hosting both replicated Repository Serer and License Server connected to a primary server (through VPN) for faster access speed (see the illustration below)
Project Subscription	Project Subscription is a signature feature of TestArchitect. By subscribing, the team can avoid effort duplication and allow a TestArchitect project to reuse all test assets (actions, data sets, interface definitions, picture checks, etc.) that already exist in other TestArchitect projects (learn more).
Result storage and sharing	Typically only one final result is stored in the shared repository for a particular revision of a test module with each version of the system under test. It's not recommended to add 'testing' or 'debugging' results into repository. Otherwise, the size of your repository will be unnecessarily large



BEST PRACTICES

Composing and Sharing Best Practices

It's always good to have centralized conventions and best practices for your team, e.g. how to break down test modules, write tests, name variables, execute tests, script review process, etc. Doing so standardizes the workflows, thereby increasing efficiency and scalability.

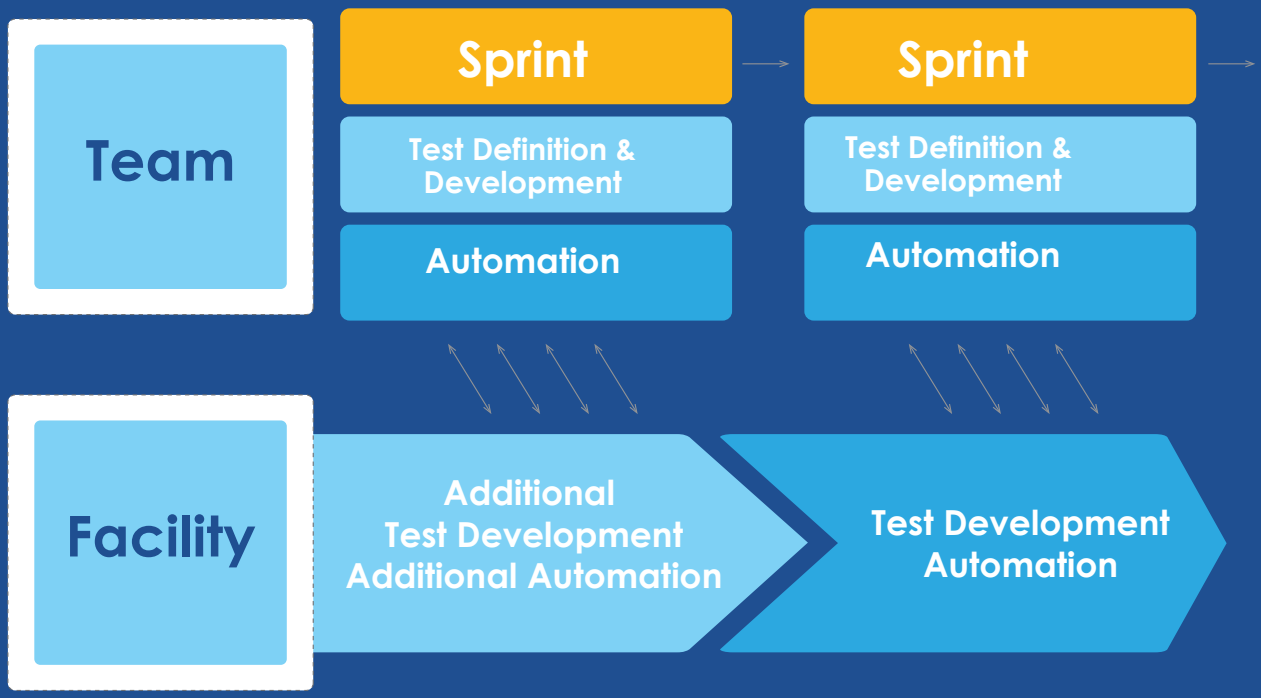
Offload to Catch up

Due to the fire-fighting pressure from the release schedule, there are times that your team needs back up by another internal or outsourcing team.

- Your team can catch up with the release cycles and get automation testing "done"
- One outsourced facility can support multiple teams, smooth out peaks and valleys
- Allow sprint teams to focus on development priorities
- Team is responsible, and decides if, what and when to outsource

Invest In Testability of the System under Test

A well-engineered framework and the best test design in the world won't amount to much if your developers don't play their part in promoting testability of the application under test. If at all possible, testability should be one of the first requirements of a new feature. In particular the developers should decide early on automation ID's for UI entities and elements. Interface definitions can then be created early on without even the need of Interface Viewer, and they will remain largely current throughout the sprint. Learn more *here*

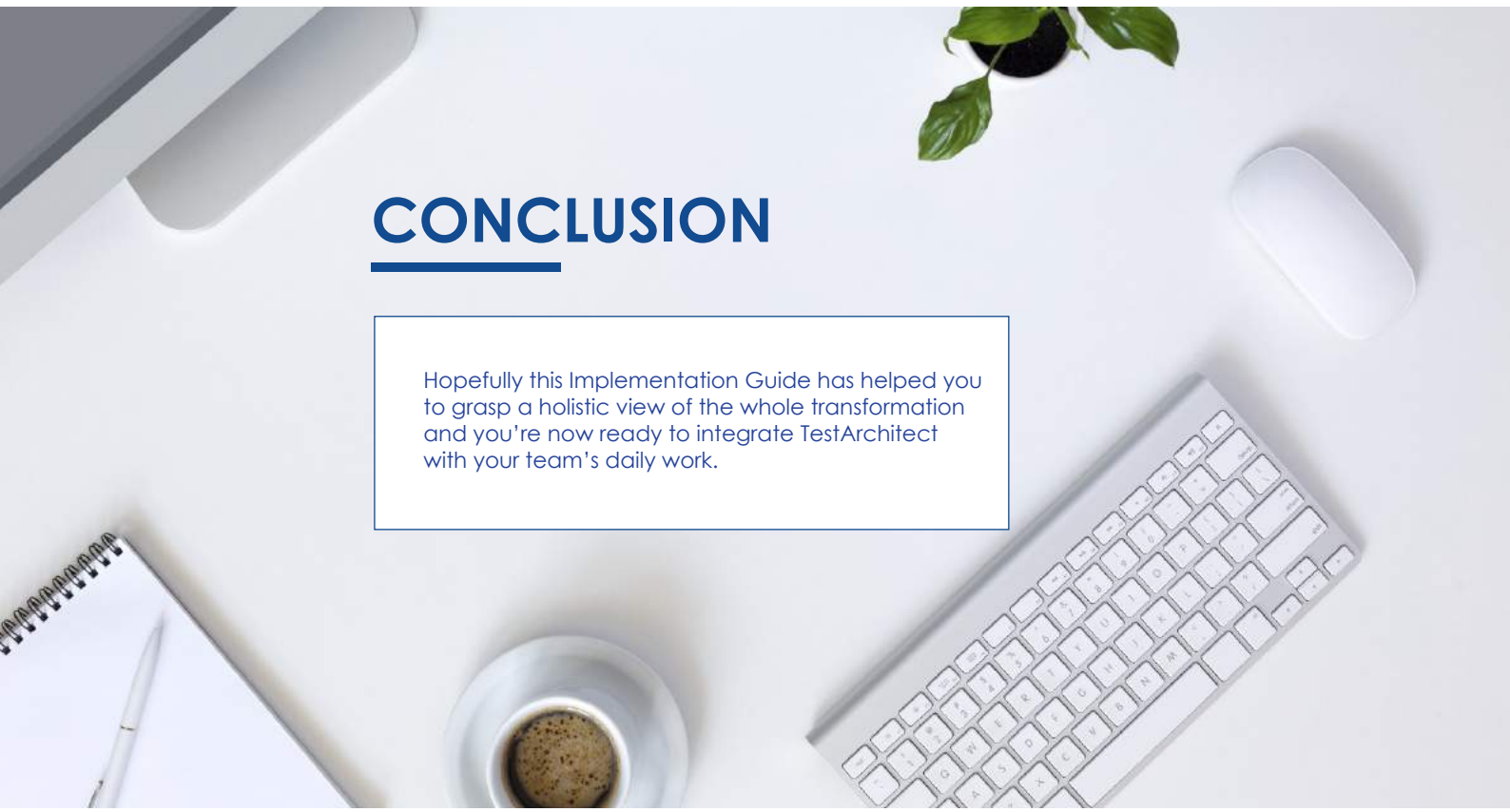


USEFUL RESOURCES

Source	Description
TestArchitect Online Help	A comprehensive user manual online containing various topics.
Video Tutorials	Get hands on with TestArchitect's video tutorials
New TestArchitect Releases	There are regular releases and updates of TestArchitect. Be sure to check them out

CUSTOMER SUCCESS PROGRAMS

Source	Description
Responsive Product Support	Our around-the-clock technical Product Support team makes sure your questions get answered promptly. Post your questions on *TestArchitect Support Portal.* You can also access our *knowledge base* containing TestArchitect's tips and tricks. Besides, you can choose among our various Customer Support services to sign up for a package which best suits your team's needs. Contact information Phone: +1 800 322 0333 Email: *support@logigear.com* URL: *http://testarchitect.logigear.com/support.html*
Training & Consulting	Our consultants review your testing objectives and processes and develop the plan to maximize your value from test automation.
Automation Delivery Services	Leverage our automation delivery services to reduce your upfront automation effort, ongoing or on-demand, to achieve the full potential of automation
Automation Extensibility Services	On-demand development services for special automation needs such as legacy UI controls, no-standard protocols, integrations with ALM & TCM tools



CONCLUSION

Hopefully this Implementation Guide has helped you to grasp a holistic view of the whole transformation and you're now ready to integrate TestArchitect with your team's daily work.