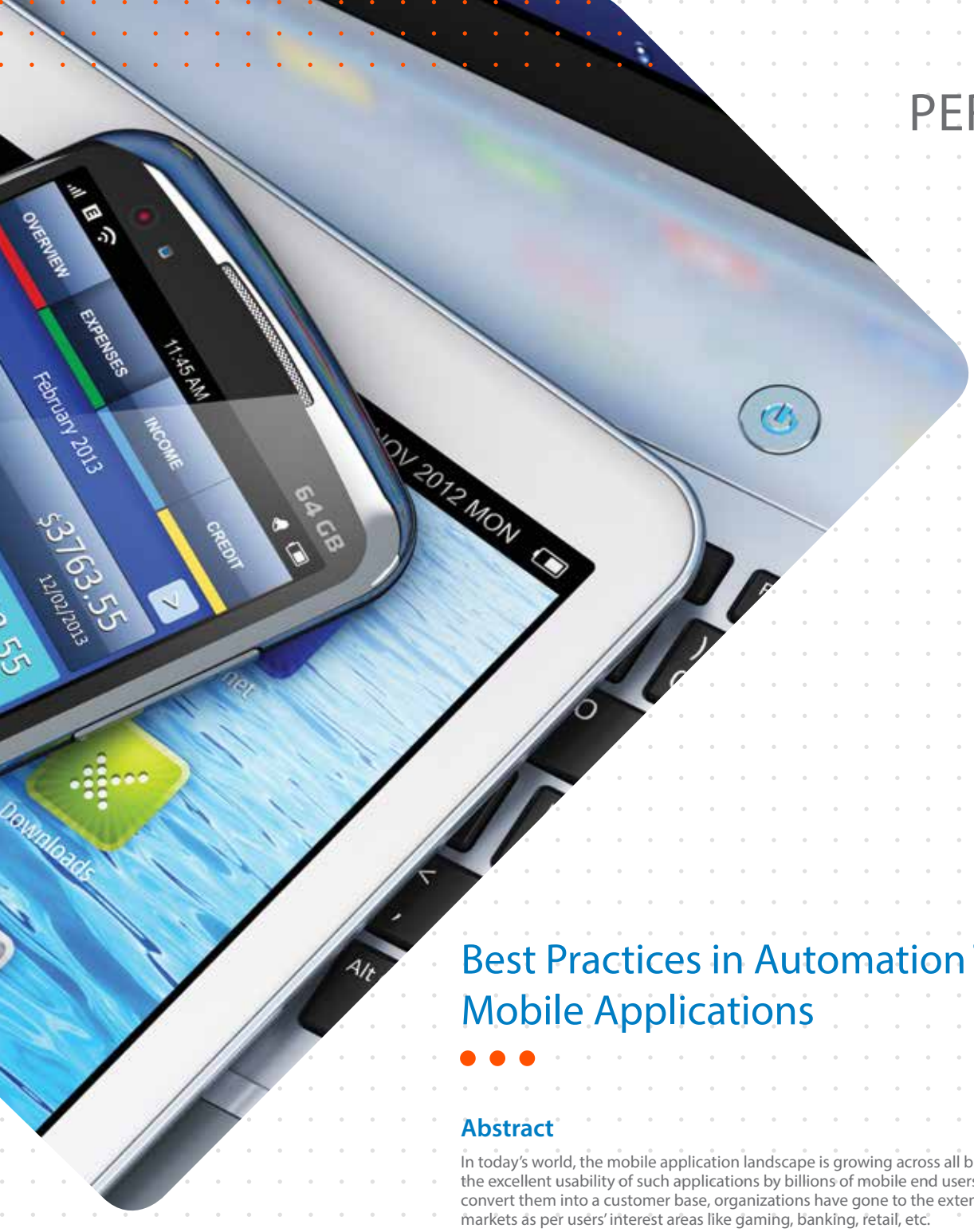
PERSPECTIVE



Best Practices in Automation Testing of Mobile Applications

Abstract

In today's world, the mobile application landscape is growing across all business verticals because of the excellent usability of such applications by billions of mobile end users. To tap such mobile users and convert them into a customer base, organizations have gone to the extent of creating their own app-store markets as per users' interest areas like gaming, banking, retail, etc.

However, this also poses challenges on the cost optimization due to the ability to support various types of mobile devices, and operating systems like Android, iOS, and Windows. QA is costlier than even development, as organizations need to

- Test-certify on too many device models, OSs, platforms, and combinations
- Do such testing more often, because of multiple OS upgrades, launch of new device models, new releases to remain competitive on mobile features, etc.

In such a critical situation, we need a solution where we can automate testing as much as possible.

Keeping in mind the most important aspect of industry demand, we are writing this paper to help understand the space, prove the value of automation, and understand its challenges and best practices.

Infosys
be more

Overview of the mobile automation space

Automating every kind of mobile functional test case is neither possible, nor required. Automation percentage will vary based on the functionality of use cases, tool selection, the type of application, and business process automation feasibility.

It is observed that while organizations have clear mandates for automation, there is an unexpected absence of a matured automation-tool strategy. The confusion over tool features and compatibility has caused organizations to start their automation journey by hunting for open-source tools first. The base reason, however, is the unclear understanding of RoI and misunderstanding the tool's true features.

Today, there are many automation tools available – like Appium, SeeTest, MonkeyTalk, and Calabash, some of which are open source.

Tool vendors have also created cloud-solution options to capture opportunities where an organization does not want to make heavy initial investments. Tools like Perfecto Mobile, DeviceAnywhere, and Sauce Labs offer solutions to automate testing and also reserve devices over cloud for nearly all kinds of QA work.

What should companies look for in mobile automation?

Mobile automation is no different from any other strategic initiative in an enterprise. If done right, it can make the enterprise very efficient in mobility QA. If taken lightly, it will not give any real benefits.

- Mobile automation will not remain isolated for very long. It will gradually integrate with multiple business channels and technical processes, to evolve towards a complete framework based on automation tools. Companies should look for –

  How one can combine web, mobile, and desktop application testing into a single strategy, with the best reuse of functional knowledge, tools, and technical innovations.

- Automation is not limited to QA activities alone. Development teams can also use automation tools for unit testing. This could potentially help prevent defects at the source, and improve the quality of applications in a very effective way. Companies should check –

  How automation tools can integrate with the development technology landscape and improve the overall development quality of the application.

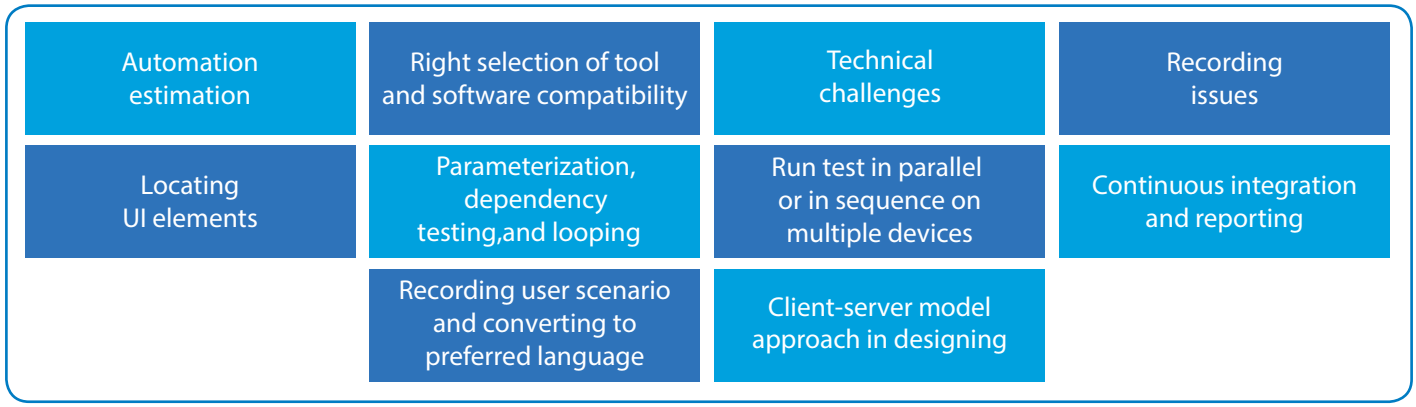
- How to ensure the ability to `automate from anywhere' so that the entire development and QA teams have access to the tools and devices to automate tests

- Any interdependency between development and QA caused by the tool itself could start becoming a bottleneck. Companies should avoid code-intrusive tools and any such interdependencies.

- Mobile QA has a significant role in ensuring content delivery and content-rendering on device. If the automation tool can also address performance testing under a single roof, it will be an added advantage.

Point of view for best practices

Mobile QA automation has areas that are prone to show up as challenges. You must understand such areas in advance, and be proactive to follow the best practices to overcome these challenges quickly.



Automation estimation

Challenges

- POCs (proofs of concept) taking more effort than expected and not closing down
- Automation is not giving effort benefits when compared to manual testing
- Manual testing is still followed as test automation did not complete on time

Best practice

- Before starting an automation exercise, at the RoI stage itself, the cost needs to be calculated based on estimation
- Do not miss any of the following estimation efforts. All of these efforts are required –
  - Environmental setup effort, device clients creation, and setting up the application for automation
  - Proof of concept and pre-validation of the most critical requirement
  - Script preparation effort
  - Script execution effort
  - Any review and rework
  - Dry runs
  - Integration with third party tools like Jenkins, CVS, etc., for configuration control and continuous integration
  - Defects logging effort
  - Reports and report analysis
- Analyze deviations in the estimated versus actual effort, as it will point to a key area not going well – such as, incorrect tool selection, unavailability of automation skills, etc. Re-plan project
- Estimation effort may not remain same for all tools
- For recording user stories, each and every tool will have its own way of working (e.g. Appium uses location of native / web elements by UI locators, MonkeyTalk uses Monkey ID for recording purpose). The tool may not have a good recording interface or elements-recognition capability. Ask the tool vendor to update how estimation effort will change based on app type and OS
- It is always ideal to understand the user base and target the automation-percentage to be achieved on the planned device / OS combination. Rely on the mobile app analytics to arrive at this data

The mandatory expectation should be that automation estimates will depend on the tool selected. Understanding of effort, based on the selected tool, will also help in selecting a best-fit tool and areas of higher effort spend (tool limitations).

Adjustments to estimation parameters:

- When selecting a tool, monitor the following features when performing a POC and adjust the estimation productivity. Select a tool that has a positive impact on RoI-based on the following parameters –
  - Recording support
  - Availability of tool knowledge documents, and training / skill development costs

- Extent of reusable artifacts utilization (e.g. same script can be used for both Android and iOS)
- Efficiencies of script executions
- Complexities of integration with continuous integration tools (e.g. Jenkins)
- Ability to capture logs / screens for defect loggings
- Reporting support provided in the tool

Selecting an automation tool

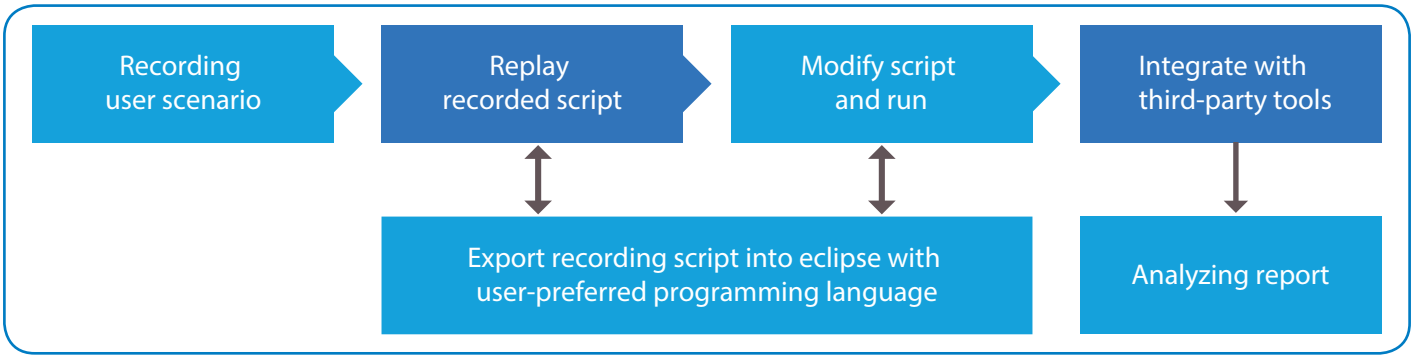
Challenges

- There are multiple tools claiming to be the best-in-industry
- Price varies from being free to very costly
- Multiple architectures (e.g. cloud-based, emulators-based)
- Not all information is available in the tool fact-sheet to appropriately understand the criteria to select
- Unknown roadmap and unclear long-term commitments from tool vendors

Best practice

Mobile automation tool selection is the most important and most sought-after area of any mobile automation strategy. While there are multiple tools, the good news is that most tools work on very similar architectures and have similar base building blocks.

Shown below is the basic working model diagram of automation tools:



Initially, record the user test case. This saves time for creating manual scripts right from scratch. After a few replays and customizations, the script gets calibrated for the expected outcome if it is run from the UI panel of the automation tool. We then export the recorded script into a user-preferred programming language to apply parameterization, looping, dependency testing, and the likes.

For selecting the right tool, take the approach / steps below –

- Create a specific task during project planning for evaluating, selecting, and performing a POC on the tool and pre-verify it. Tool selection is critical. An incorrect tool will cause investments to go down the drain and result in missing project timelines and more effort / cost.
- Also collect statistics on the effort spent, technology skills, customization required, etc., during the tool evaluation phase, to recheck RoI calculations.
- Do not plan isolated projects; list all mobile application types and OSs (including desktop app OS, not just current app but also planned future apps) used in RoI calculations. All tools do not work on all mobile application types and OSs. The tool vendor will be explicitly detailing out all compatibility issues.
- Avoid tools in which a programming interface for script-modifications is not available, as most of the times we need to make changes in user scenario and reporting.
- Select a tool that will enable execution on both devices and simulators. Most developers rely on simulators to verify functionality, and if we were to move validation / quality checks to earlier in the release cycle, this will be useful.
- Community support for the tool is another major criterion which should influence a tool selection.
- Page object pattern is the best choice for writing automation scripts and has benefits in script reusability. It will also make the test more readable, maintainable, and robust in an Agile environment.

The table below provides a quick general / illustrative compatibility of current automation tools with mobile app types / OSs

Tools	Android			iOS		
	mWeb	Native	Hybrid	mWeb	Native	Hybrid
Appium	Y	Y	Y	Y	Y	Y
SeeTest	Y	Y	Y	Y	Y	Y
Silk Mobile	Y	Y	Y	Y	Y	Y
Robotium	?	Y	Y	NA	NA	NA
iOS Driver	NA	NA	NA	?	?	?
Frank (iOS)	NA	NA	NA	NA	?	NA
Android UI Automator	?	Y	?	NA	NA	NA
Seledroid (Android)	?	?	?	NA	NA	NA

NA - Not Applicable **?** - Required Attention for POC **Y** - YES **N** - NO

Quick data sheet on a few popular tools

Tool name	Licensing type	Recording	Object identification
Apache JMeter	Open source	Y	N
SeeTest	Medium price	Y	Y
Selenium, Appium	Open source	Y	Y
MonkeyTalk	Open source	Y	Y

Tool name	Web/mWeb	Native iOS	Native Android	Native Blackberry	Native Windows
Apache JMeter	Y	N	N	N	N
SeeTest	Y	Y	Y	Y	Y
Selenium, Appium	Y	Y	Y	N	N
MonkeyTalk	Y	Y	Y	N	N

Tool name	Same script runs on all platforms	Remote device testing	Parallel run on multiple devices	Log capture for defect logging	Screen shot capture for defect logging
Apache Jmeter	NA	NA	NA	Y	N
Seetest	Y	Y	Y	Y	Y
Selenium, Appium	Y	Y	Y	Y	N
Monkey Talk	Y	Y	Y	Y	Y

Tool name	Out-of-the-box integration with test mgmt. suite	Integration with CI tools	Improved result reporting support
Apache JMeter	Y	Y	N
SeeTest	Y (HP QTP, Sel., JUnit, VS, and Python)	Y (HP, ALM, and a few others)	Y
Selenium, Appium		Y	Y
MonkeyTalk	N	Y	N

Handling challenges during recording

Challenges

- Replayed script not working; recording takes more time and effort. We end up applying test / image validation, page load and element waits, and correct unique element ID
- Recording not supported for RWD, native, or hybrid mobile applications
- Application crashes during recording
- Application times out

Best practice

- Check explicitly with tool vendors the recording features provided by 'Application Type' and 'OS' (e.g, mWeb app for iOS testing). A few tools may not support recording for RWD web-apps the way they do for native apps and scripts to be written manually.
- For web-apps, check with the development team if the application is fine-tuned for mobility. Memory and CPU usage not tuned for mobility could cause applications to start crashing while recording.
- Check tool connection options with the device (USB / IP Address / UDID, WiFi). Connectivity plays a role in speed, and good infrastructure needs to be ensured. Tools that use JSON-wired protocol use USB connected devices.
- If the tool provides recording facility but is not working, check element XPATH / ID and replace correct ID / XPATH in recorded script.
- If the tool does not provide the recording interface, try Selenium-provided recorders, specifically for RWD mobile apps. We will need to write the script manually for each element corresponding to its locator ID and that will take effort. Estimate the effort accordingly.
- While exporting the recorded script check if there is any missing library to be added in the build path.

Location UI elements

Challenges

- Technically, for automation, one will need to recognize the application's unique ID to determine whether an object is text / image that can be located on the page
- Huge effort required to debug and fix, as the tool is not able to find location UI elements or discovers too many elements with the same ID
- The tool fails to locate specific elements on a page.

Best practice

When searching for an element on the page, tools behave differently with different mobile application types.
Try avoiding a tool that does not support both recording as well as UI elements identification (even if it is free of cost). It is an effort intensive task and the costs saved from the tool will go into manual effort.

- There are alternative solutions possible though - e.g., Appium automation for RWD web apps can use Chrome browser mobile emulators for locating element of a responsive web page
- If application is hybrid / native, then for Android application, Android SDK in-built UIAutomator can be used for element ID. iOS application can use tool-provided locators like ObjectSpy (SeeTest), component tree (MonkeyTalk), and Appium Inspector (Appium)
- Chrome and Safari developer tool options have locators available; there is also an option for copying XPATH / CSS PATH for any element locating on the web page
- MonkeyTalk uses the Monkey ID for locating elements on a page, or using component tree we can locate elements on a mobile screen
- Tools can timeout while identifying objects. Tools using socket programming might timeout at the back-end when they are unable to differentiate between native and web elements
- It is advisable to have a Page Object pattern approach for UI testing (especially when there is a possibility for UI changes) using a multi-tier approach. It will provide very easy and maintainable scripts in the long run, that is, Infosys IP ZED3 and iWAF

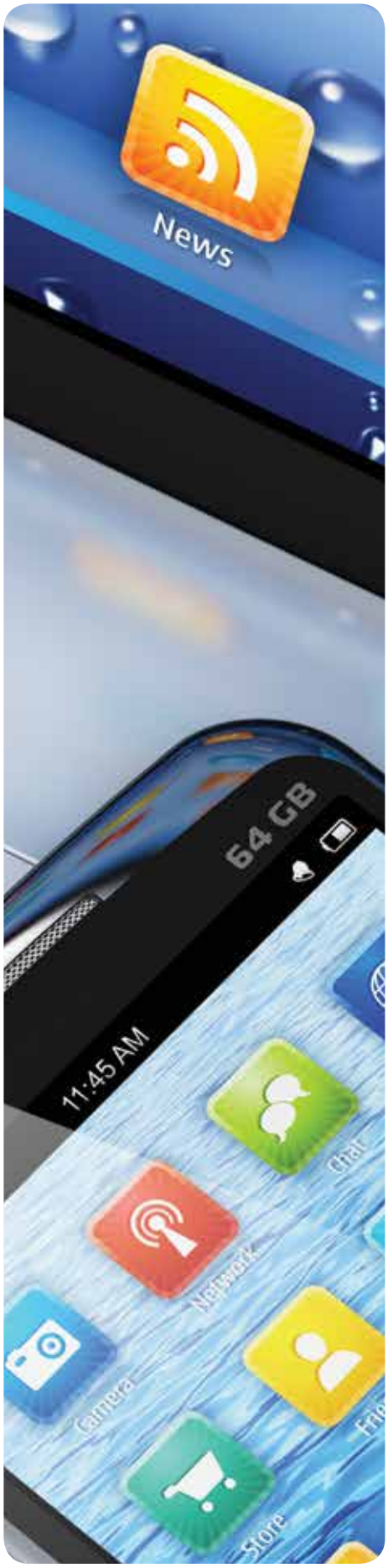
Handling Environment setup issues

Challenges

- Installing a tool requires library knowledge
- Environmental setup will need understanding of operating system environment variables, terminal commands, and path and classpath setting for virtual machines
- Environmental setup varies based on mobile application type as sometimes we need instrumentation code, and some vendors provide instrumented app Browser that can be used to connect and record

Best practice

- Plan skilled automation resource or trainings in the project plan itself
- Note down all information like Apple developer account license, mobile devices UDID and server machine required with software components (XCODE Vx.x, Android SDK Vx.x, etc.)
- Sometimes, the latest version may not be the right choice for all dependent components. Do not install just the latest versions. Explicitly check all dependencies on components versions.
- Prepare to debug application code for any issues like instrumentation, crashes / app is not getting installed on mobile device, etc.
- Get administrator rights on machine as you will need it
- Always have the set-up scripted out using tools like Chef or Bundler (Ruby) if possible; this will ease out the effort spent on setting up a new machine for automation



Parameterization, dependency testing, and looping

Challenges

- Parameterizations for consuming dynamic values cannot be avoided. Plain text, Excel sheet, XML, or some random generators are obvious options
- Scripting is sometimes difficult, as it requires API knowledge and programming skills on specific platforms
- Dependency testing (one test depends on other test conditions and hence needs looping) needs recording, and export into other languages. Tools with low recording support will require significantly higher effort and skills

Best practice

- Use framework supported API available for parameterization, looping, and dependency testing, that is, TestNG
- It is recommended to use looping even if the tool does not provide much support. Looping will shorten the length of the script and make it easy to understand. We can also change the script data without any change in the script for any parameterization and dependency testing
- Use multi-tier approach with best design pattern suited to application

Run test in parallel or in sequence on multiple devices

Challenges

- Usually considered advanced level and neglected
- Requires detailed understanding and the tool's support-limitation for multiple devices with multiple OS, to factor in the extent, and levels of compatibility to be included in the script

Best practice

- Pre-understand the support provided in the tool, at the time of tool selection
- Client and server approach: Most tools support client and server architecture. Do not take a corner and install automation client and server both on single machine. Advantage of this architecture is –
  - Devices are connected on server machine and multiple automation testers can work in parallel from client machines.
  - It saves RAM eaten by the automation server that slows down the machine automation engineers working on it (if server is installed on the same machine).
  - Server configurations are tightly controlled
- Please note that the client and server machine should be on the same network, without firewall restrictions
- It is suggested to use a MAC machine as a server machine, as it can be used in both Android/iOS automation
- UDID is a simple mechanism to identify a specific device. IP address / Device Name can also be used sometimes without any harm
- Tools will usually support recognizing devices connected through USB, using ADB.XCODE for Android / iOS devices. Additional scripting will be required for executing tests in parallel on multiple devices
- Some automation tools have instrumentation limitations on specific platforms. For example, Appium can have only one iOS device used at a time in automation run, but multiple Android devices can be can be run in parallel
- Sequential running test on devices can be controlled from the priority given to methods

Continuous integration and reporting

Challenges

- Continuous Integration (CI) will require additional setup, and talent with CI tool configuration knowledge.
- If CI is not planned and ignored in the planning stage, it can impact project dates. It will take some time to get integration working and eventually will need fine-tuning.
- Every individual application might need a separate continuous delivery pipeline setup.

Best practice

- It has to be planned in the project plan itself. CI is the most desirable feature as one can achieve automated builds, schedule automation run, single click status, set the duration of automation test run, get failure result summary and automating email notification for each run, etc.
- Utilize the Infosys centralized web-based Continuous Integration Platform (ICIP) that hosts a standard set of tools, right from version management, to static code analyzers, unit testing tools, CI server, and the defect tracking and test management tools. This platform helps in end-to-end automation through integration of various open source / licensed CI-CD tools. It also helps to achieve automated code review, monitor test coverage, and measure quality metrics.

Additional tips

- Tool skills will need to be internally developed. Select a tool that has good support and documentation available. Most open source tools do not provide good documentation or support. This will increase overall TCO, even with free tools.
- Some automation tool environmental setup requirements are difficult, such as, 'Appium' setup for iOS on real device. Though the tool is excellent, initial hiccups could cause timeline issues.
- It is really not suggested to perform tool selection and POC in the middle of a project as it can impact client deliverable and cost.
- Combine functionalities which are dependent on each other, such as the Home and Login page.
- Differentiate between native and web element by tool supporting; for example, on RWB web apps you can get native pop-ups such as "do you want to translate this page into English?"
- Try running an independent flow for each sequence to avoid other flows running in the same test.
- Do not re-insert USB between an automation run to avoid abnormal/ illegal program termination.
- Use parameterization and data provider concept for running a script multiple times with different URLs each time.
- Plan to have a 'think-time' via Selenium wait - to avoid element wait or timeouts
- Use a good 3G / 4G data network to avoid test failure due to network.
- Keep the device and server machine in `unlocked' mode.
- Log each step result for better debugging and reporting.
- Put suspicious code into try / catch block to avoid abnormal program termination.
- It is better to use Eclipse environment with your favorite programming language to make scripting more readable, effective, dynamic, etc.
- Sometimes clicking an element on a page may not work because the element is not present in an active area, so we need to scroll and click on that element. Even if the click is not working, you can program a JavaScript click on the element when it does not respond to the Selenium click. When we need to look into how the code is behaving, this click can be the basis for finding the unique and correct element id.
- Divide and run the script in parts if there is an issue in reporting the complete script at once due to network / data.
- Always having the try / catch block might not be a good practice. It is always good to have wrappers written around and errors raised with custom messages with the actual message appended to it. Having screenshots embedded in the reports is very helpful while analyzing failures to determine if they were legitimate or not.
- It is advisable to use BDD (Behavioral Design Development) approach of system, rather than TDD (Test Driven Development) approach, because BDD drives customer interest and focus, something non-technical stakeholders can also easily collaborate upon.
- Before executing test cases, device, and application state and feasibility need to be checked.
- It is good to write independent flow and stateless test cases.

Conclusion

Effective automation testing using tools as per the selection criteria and budget can be achieved by using the testing strategy and best practices described in this paper.

Several factors like availability of tools and compatibility, network and environment setup, recording and location elements on UI tricks, system configuration for desktop and server machines, are considered for effective and on-time automation project delivery that earns client satisfaction and appreciation.

About the Authors

Manish Kumar Chauhan

Consultant, Digital Customer Experience unit, Infosys

Manish Kumar Chauhan is a consultant with the Digital Customer Experience unit of Infosys. He has more than eight years of total experience in software development and testing. He has extensive experience in manual, white box, API, unit automation, and performance testing on web applications, mobile devices, set-top boxes, digital TV, security devices, etc. Manish was involved in creating test cases, test plan, test strategy, automation framework, automation script, and performance script. Manish has worked on many automation tools like Appium, SeeTest, SilkMobile, and MonkeyTalk.

You can reach him at **manish_chauhan01@infosys.com**

Vivek Rai

Senior Project Manager, Digital Customer Experience unit, Infosys

Vivek Rai is a Senior Project Manager in Digital Customer Experience unit of Infosys. He is currently handling multiple clients in mobile testing area, providing a range of services, including manual functional, UI compatibility, and automation and performance testing. Vivek started handling mobile-driven projects as early as 2007 and since then has been continuously involved in mobile development and quality assurance. Beside technical, he has extensive experience in projects planning and roadmap creation, mobile consulting, communication management, team management, presales, etc.

You can reach him at **vivek_rai @infosys.com**