TADEK – Test Automation in a Distributed Environment

A novel approach to automatic testing and remote control of applications over accessibility and similar technologies
What will you learn?

- The importance of automated end-user testing
- How TADEK can solve quality assurance problems in small and large scale projects
- The main features of TADEK
- How TADEK controls applications without a single change to their code
- Some details about the architecture of TADEK
- Basics of models and test cases
- Different user interfaces and their usage for exploring applications, testing and continuous integration
- How to scale TADEK’s environment and run tests on many devices in parallel
- How COMARCH can assist you with TADEK deployment or any Quality Assurance issues
Introduction

Quality control is a time and resource-consuming task which is quite often neglected. Many people think testing adds no value to the project; no new functionality is designed or implemented in this phase. It is tedious work and the results are virtually invisible for the end user. However, everyone agrees that quality matters. Nobody would use even the most sophisticated software if it crashed often or behaved out of sync with the user’s expectations. It is clear that no serious project can succeed without thorough quality checks.

Like it or not, quality assurance usually costs a quite significant part of a project budget, especially when executed manually. Manual work is priceless when taking into account exploratory tests, but forcing people to run the same set of test cases after every build is just a waste of money, people’s knowledge, and experience. Not to mention that after a few weeks of such a repetitive task even the most dedicated engineer would start searching for another job.

One can say that there are unit tests which seem to be quick and solid solutions to those problems. Every function in the application code is tested and proven to work according to the design. However, for some reasons when all those functions are put together something bad happens and the end user experiences problems not found in low-level testing.

What happened? How to prevent such situations? How to track them down as early as possible to minimize bug fixing costs?

Fortunately, there is one answer to those questions – automated end-user testing. This approach imitates end-user interaction with an application interface, just as if it were being used by a real, human operator. This seems easy, but how to implement such tests? Build a robot to click on the screen, and camera with artificial-intelligence software to read the resulting content? Well, sometimes it is necessary but in most cases the same can be achieved with much simpler, faster and cheaper tools.

TADEK – Test Automation in a Distributed Environment – is a new approach to solving this problem and provides the means for automated end-user testing of applications, without additional effort during implementation. TADEK is a free solution, soon to be open-sourced, based on accessibility and similar technologies, providing all that is required for exploring the content of applications, writing test cases, running them, and reporting the results. Application models and test cases are written in Python, making it simple to learn and easy to extend. Client-server architecture allows parallel testing on multiple devices, and drastically reduces resource consumption on tested systems, which is always an issue for mobile devices or performance tests. The tool is also extremely user-friendly, due to the Qt-based GUI. However, a command-line interface is also available, and can be used in continuous integration systems, such as OBS.

The main goals of the tool are:

- Providing a means for quality assurance of every project, even on a small scale
- Simple and rapid test implementation, with no change to the application code, in order to minimize the extra costs of QA
- Unified extensibility for provision of one common interface to many access technologies
- Maintainability for re-using the same test cases on different projects and platforms
- Flexibility to integrate with other systems, such as continuous integration

TADEK is not just an automated testing tool – it is a full testing environment. Apart from collecting and running test scenarios and assessing their outcomes, it also watches for core dumps in case of crashes happening during testing. It can use any native system command or script as a part of a test scenario. The results are sent through output channels built as plugins, so it is extremely easy to integrate it with other tools, for example test management or reporting solutions.

TADEK in small and large scale projects

Regardless of the size of the project, the cost of quality assurance is always significant. TADEK for end-user testing helps to reduce the cost, in both money and time. Thanks to the access technologies on which TADEK is based, there is no need to add any extra implementation to the application code. Control of the application, and reading and assessing its output, are performed without a single change to the application itself. Designers and developers can focus on implementing functionality and not waste time on creating hooks for test tools. Moreover, one can perform tests on any application without access to its source code. There is no need to re-compile the application with extra libraries - all that is needed are the same binary executable files as used in a real-life scenario.

TADEK is a Python-based tool. Its engine is written in Python and the test cases are Python scripts. This approach gives several advantages:
Python is very easy to learn so there is no need for expensive training for test case developers.

If you know Python there is no need to learn another syntax to write test cases.

It is simple to create extensions and add-ons.

It is an interpreted language – just write your test cases and run them, no need to compile, link etc.

## Multi-application scenarios

Virtually every device now on the market is capable of running multi-tasking, multi-application processes. It is typical to browse the Internet while writing mails or messages to friends in social networks, with favourite songs playing on the music player application. With TADEK you can automate such scenarios even when all those applications come from a different vendor. It is easy to test interoperability – for example opening a mail application after clicking a mail address on a web page in a browser, filling out its content, sending it, then checking new mail in one's mailbox; or opening a URL in a browser from a message sent from a friend through SMS or instant messaging application. Or any other scenario you can imagine – just name it.

## Accessing the applications

How is it possible that TADEK can control applications without injecting any code? Applications, especially those built with Qt or GTK+, consist of widgets like buttons, menus, labels etc. If you can access the widget you can access any application which uses this widget.

Technology called ‘accessibility’ (often abbreviated to a11y) focuses on exposing systems’ and applications’ functionality and content to people with disabilities. It provides interfaces which enable full access from other software or hardware, like screen readers with speech synthesis, Braille, magnification, etc. Accessibility is standardized by Open Accessibility Group, part of the Linux Foundation, so all interfaces to applications and services are unified even across different computing platforms. This feature also makes it an ideal tool for automated testing.

TADEK uses accessibility and similar technologies to access application UI and content for automated testing, or merely remote control. Some foundations are similar to Dogtail, which also uses any, but the actual approach is very different, and functionality is far wider. Even though accessibility is widely accepted as a standard, not all widgets and application content elements have its interfaces implemented. For example most web browsers do not expose page content as a set of accessible widgets. Due to that limitation, TADEK was designed to use virtually any possible way to control application content through access plug-ins. Hence it is not limited to accessibility; currently available plug-ins can test web page content through Document Object Model (DOM), Mozilla’s browser user interface through XUL, and a separate plug-in for WebKit is also available.

TADEK has a separate middleware layer between the engine and the technology, used for accessing application content. Test developers do not have to learn each technology to write test cases – TADEK provides a standard API, the same for every technology, and all tests are written using just this API. It is the middleware which translates TADEK’s API to the actual technology. Moreover, different technologies can be used together, in one test case. Furthermore, the user can create additional layers, and utilize them in a similar way to those already available. For example one can imagine accessing some content just by taking screenshots and analysing them when clicking just on X-Y coordinates. Seems useless, but once implemented it can be used for any application, regardless of the framework used to implement its user interface.

## Architecture of TADEK

TADEK has a built in client-server architecture, which means, among other things:

- parallel testing of multiple devices at once
- lower memory and CPU footprint
- user-friendly, Qt-based GUI or command-line interface used remotely from any PC, or locally through a localhost connection

On the platform under tests there is just a small daemon running as a server. Its only purpose is to execute commands giv-
by the client through specified access technology, hence it is very light. It is the client that controls test execution. TADEK graphical user interface, a stand-alone application written with Qt is its main client. There is also a command-line interface client providing similar functionality as the GUI which can be, for example, used for combining TADEK with continuous integration tools.

The server and the client communicate over TCP/IP using XML-based protocol. Usually the client works on a separate PC – especially in case of mobile device testing when minimal resource consumption is crucial. It is also simply more convenient to use a big screen to control testing process. However, there is nothing preventing the client from being run directly on the tested platform. This is pretty common when using a command-line interface client.

if currently available plug-ins are not satisfactory. Users can contribute their plug-ins to the community so the catalogue of plug-ins will grow over time. When this text was written there were several plug-ins available to control applications, for example:

- Accessibility for Qt
- Accessibility for GTK+
- DOM for Mozilla browsers
- DOM for WebKit
- XUL for Mozilla browsers

There were also several plug-ins for output:

- XML files
- HTML files
- Standard output
- Remote logging through TCP/IP

Parallel testing and scalability

TADEK clients can connect many servers at once and run tests on all connected devices in parallel. Neither extra equipment nor software is necessary – just connect more devices to one client and run your test cases. TADEK will control all of them at the same time, taking care of sending proper commands to the proper device. In this way testing can be significantly accelerated – the more devices you connect the faster tests are executed. And there is no limit on the amount of connected devices, which makes TADEK an extremely scalable solution - for example for those having a huge base of test scenarios to be run with every integration round.

Plug-ins

As mentioned above, TADEK is very modular. It uses plug-ins for both accessing applications and creating output with the results of executed tests. That is one of its strongest features – users can enhance TADEK capabilities to serve their needs
Test cases

Test case definition relies on the model of the tested application (or several models in case of a multi-application scenario). It is a Python script describing which action should be performed on which widget, and how to check the results with assertions. This is usually divided into steps. These are atomic logical entities providing results so each step may successfully pass or - in case the application behaves other than specified - fail. It is typical to define test cases in a way that one failed step causes failure of the test case, but this is not obligatory.

Test cases may also contain additional descriptions which can be used for different, even user-defined, purposes. The most common way is describing assertions to provide more descriptive information about failed steps. Some descriptions can be used to simplify co-operation with continuous integration systems such as OBS.

TADEK GUI and command-line interface

TADEK was designed to be easy to learn and use. This would be impossible to achieve without a user-friendly interface. It is also developed in Python with Qt. Both of them are available on multiple platforms, and on all those platforms TADEK GUI can be used. TADEK GUI contains everything users need to create a testing environment, execute test cases, and analyse and report their results. Users can browse widget trees in every access technology, and can interact with applications controlling them remotely right from TADEK GUI. Another view provides all means to execute test cases. The next is a powerful result viewer, so the user can quickly analyse the cause of failures found during testing and file bug reports. Results can be stored and analysed later in exactly the same way as they if they came from a current testing round.

There are tools for exporting data to other applications – for example full support for HP Quality Center. It is enough to create test cases once in TADEK, then users can import them directly into Quality Center. All the results from tests can be imported the same way, drastically saving the time and effort expended to store data in Quality Center.

For those who prefer command-line interface, or for easy integration with other tools or just shell scripts, TADEK also provides this kind of client. It is fully functional the same way as the GUI client.

Continuous integration

Continuous integration is crucial for the smooth development of complicated systems. It also helps to improve the quality of software by applying quality control frequently, after integrating small amounts of fixes or modules. There is no place for manual testing in the continuous integration process. It is typical to use unit tests instead. Unfortunately; relying just on unit tests usually fails as end-user scenarios are more complicated; sometimes there may also be design and architectural flaws which are not likely to be found in unit testing. TADEK provides a neat solution to this problem, enabling end-user testing on every integration round, on every stage of the development process. Each new feature can be tested the same way a real user would use it. Each test can be re-used countless times, checking any possible regressions while developing other features or during bug fixing.

TADEK and community

TADEK will soon be open-sourced under LGPL license. It has a quite extensive road map, even though it is now a fully functional and mature tool. We are looking forward to seeing community involvement, both in using the tool, providing feedback and in implementing new features. All necessary information can be found at www.tadek.comarch/comarch.com, all sources should be available on http://gitorious.org/ soon.
COMARCH services

TADEK provides comprehensive and free QA solution for most projects just out-of-the-box. In some cases, however, users may look for professional support. The TADEK team at COMARCH, initial authors of the environment, will be glad to help in such situations. We offer, among other services:

- TADEK deployment in the client’s environment
- Model and test case development
- Creating new access layers in case of special needs in non-standard projects
- Providing new output channels according to the client’s requirements
- TADEK integration with third-party software (e.g. for test management, continuous integration, reporting)
- Test execution and reporting

COMARCH also provides Quality Assurance outsourcing, with flexible set-up and affordable pricing. Professional QA engineers ISTQB-qualified and with years of experience in providing comprehensive quality solutions internally and for many renowned vendors, are at your disposal. For more details please contact:

Dariusz Filipski
Mobile Solution, R&D Director
Comarch SA
Dariusz.Filipski@comarch.com

Mariusz Lasek
Business Development Director
Comarch SA
Mariusz.Lasek@comarch.com
Comarch is a leading Central European IT business solutions provider specializing in forging business relationships that maximize customer profitability while optimizing business and operational processes. Comarch’s primary advantage lies in the vast domain of knowledge accumulated in and applied to our software products. These products incorporate highly sophisticated IT solutions for businesses in all vertical sectors. Comarch has a multinational network of offices employing over 2800 highly-experienced IT specialists in Europe, the Middle East and the Americas.