EBOOK

Unit Test Automation

A step towards agile delivery and DevOps in SAP®

basis

DevOps + Testing for SAP

Digital changes everything

A seismic shift in the way companies operate and engage with customers is occurring. It's being driven by a perfect storm of technological development. And it's changing everything.

In another ebook — 8 steps to daily SAP releases — we saw that born-digital companies are setting the pace when it comes to updating their customer engagement systems. And traditional companies must transform themselves if they are to compete successfully.

The same ebook also demonstrated how DevOps — a software development and deployment methodology that emphasizes collaboration, communication, mixed discipline teams and, very importantly, automation — can be successfully applied to SAP environments where stability and availability have traditionally been prized above everything else. As a result, organizations can deliver frequent, safe SAP releases and level the playing field with their born-digital competitors.

EBOOK

How to Achieve Safe Daily SAP® Releases

Accelerate the pace of SAP change to support your digital strategy

basis

Download ebook ❯

DevOps is based on a number of ‘continuous’ processes, including...

Continuous Integration
Continuous Delivery
Continuous Deployment

Continuous integration automatically executes tests every time the software is changed, in order to verify its quality

Continuous delivery automates the build process that delivers deployable software to operations updates

Continuous deployment automates the delivery of new functionality to users

Automated unit testing is imperative for DevOps

Underpinning these components is automated testing, which is where significant gains can be made both in terms of accelerating development and reducing costs. And these gains not only apply to companies that are using DevOps, they are equally applicable to organizations using agile development or more traditional methods.

“Automating testing ensures quality delivery at speed. Manual testing never did keep up with the pace of delivery, and the problem gets worse when delivery speed increases. Automated testing, built into the continuous integration process and driven principally off APIs, gives developers the fast feedback they need to improve quality. By the time the application is deployed it is thoroughly tested, making release decisions simple.”

Kurt Bittner, Forrester

Source: Forget Two-Speed IT; DevOps Enables Faster Delivery Across The Board Vision: The Modern Application Delivery Playbook by Kurt Bittner, Forrester, November 6, 2015.

The importance of testing

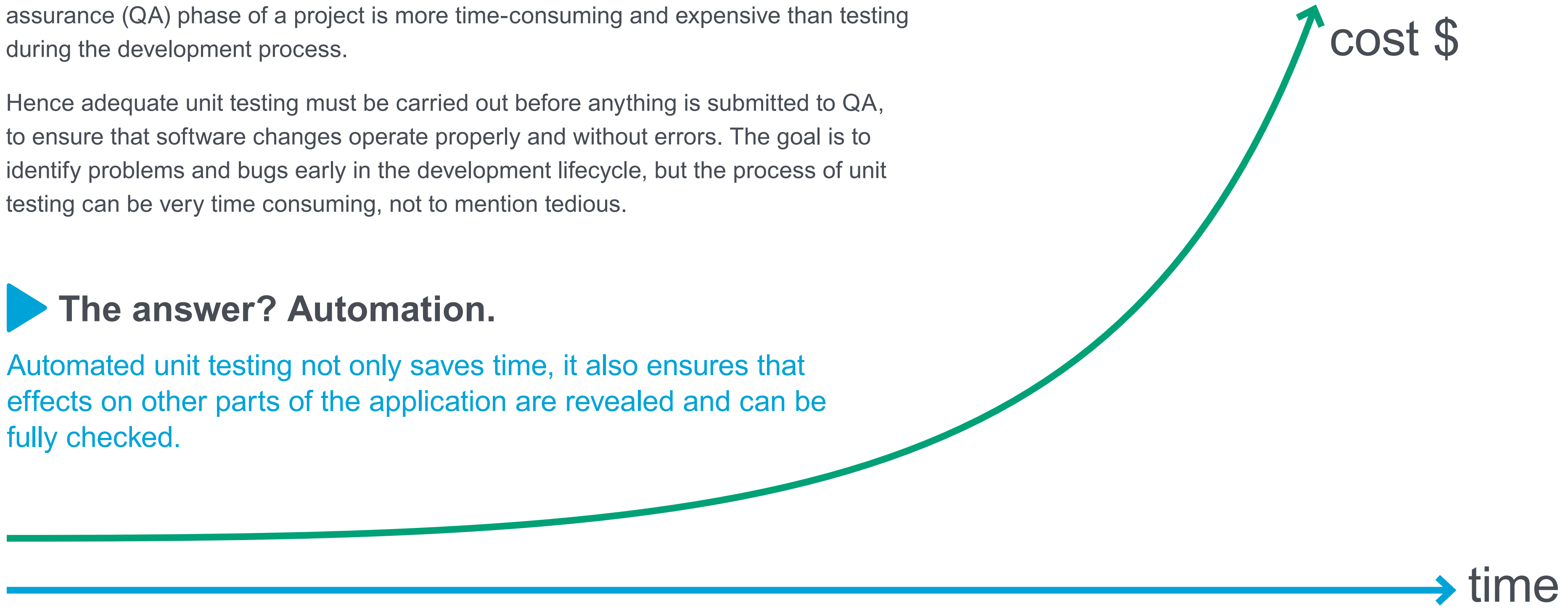
For many years it has been known that the true cost (and time) of fixing software defects increases exponentially as a project progresses. For example, a defect discovered once software is put into production is reckoned to cost one hundred times more to fix than one discovered in the development phase.

It is also true that integration testing and end-to-end testing carried out in the quality assurance (QA) phase of a project is more time-consuming and expensive than testing during the development process.

Hence adequate unit testing must be carried out before anything is submitted to QA, to ensure that software changes operate properly and without errors. The goal is to identify problems and bugs early in the development lifecycle, but the process of unit testing can be very time consuming, not to mention tedious.

The answer? Automation.

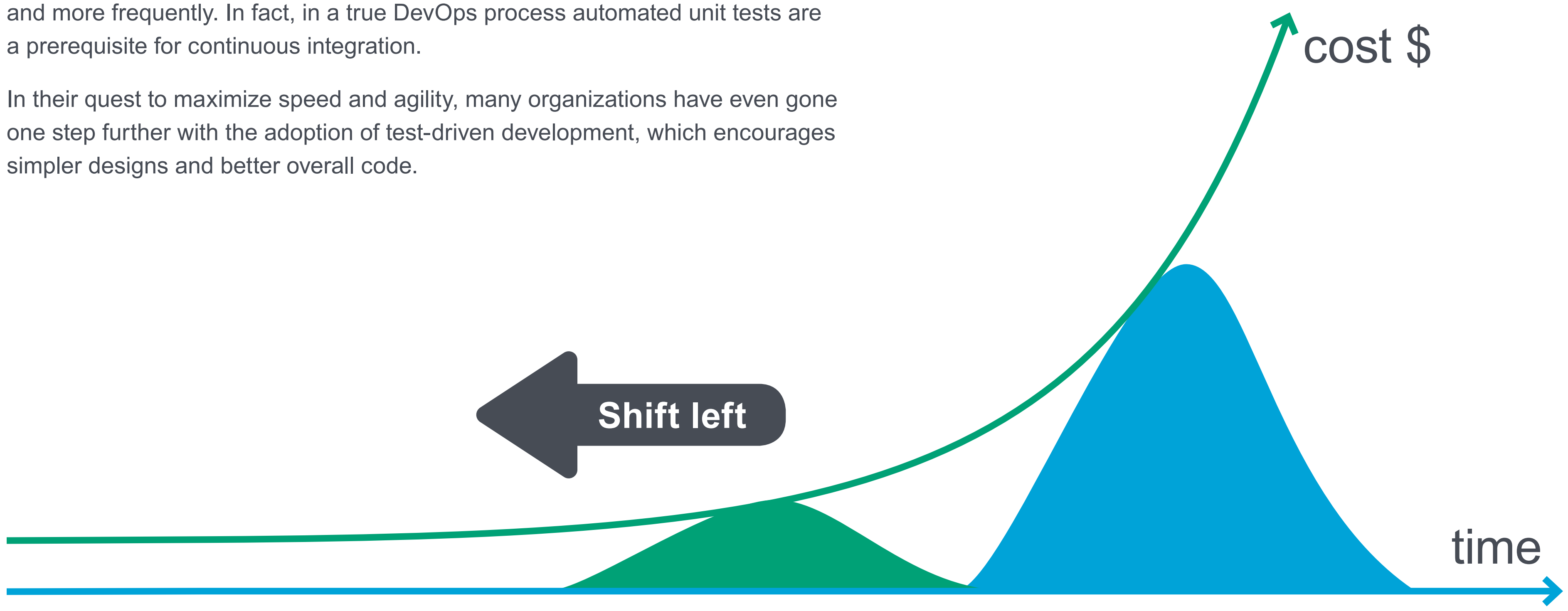
Automated unit testing not only saves time, it also ensures that effects on other parts of the application are revealed and can be fully checked.



Shifting testing as far left as possible not only reduces costs massively, it accelerates development significantly

The message is clear: whether or not organizations are following a DevOps methodology, automated unit testing allows them to deliver quality code faster and more frequently. In fact, in a true DevOps process automated unit tests are a prerequisite for continuous integration.

In their quest to maximize speed and agility, many organizations have even gone one step further with the adoption of test-driven development, which encourages simpler designs and better overall code.



Unit testing and test-driven development go hand-in-hand

There are some key principles that need to be considered when writing unit tests, in order to deliver a successful outcome.

- **Independence** – when multiple instances of the same test are run in parallel they should deliver the same results. To make sure that one failing unit test doesn't affect others, you must make sure that each can be run independently. The power of unit tests is that they will clearly indicate the source of any errors and even tell you how to fix them.
- **Consistency** – a test should deliver the same results when it is re-run (all other things being equal). Results should not depend on the environment or external data.
- **A single unit of work** – it's important to test small parts of code that have distinct functions. The narrower the test scope, the easier it will be to identify and resolve issues.
- **Good code coverage** – the test needs to ensure that all the code is executed, including exceptions.
- **Run fast** – if the test fails the developer needs to know quickly so that it can be fixed or backed out of a release.

Test-driven development goes hand-in-hand with unit testing, and in doing so turns the typical development process on its head. It's a robust way of designing software components, or units, interactively so that their behavior is specified through unit tests.

“Automated unit testing is the plankton of change in DevOps — without it everything else falls apart.”

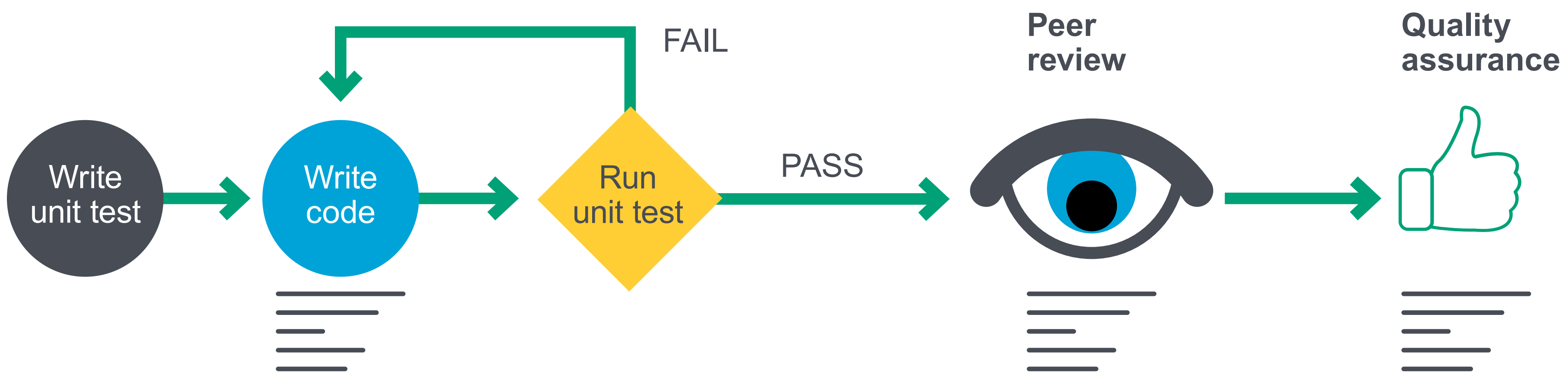
James Roberts, Chief Technology Officer, Basis Technologies

Before any code is written a unit test must be defined

As the name suggests, test-driven development mandates that a unit test must be defined for each piece of code before it is written. Crucially, in order to do this the a developer must clearly understand any new features, specifications and requirements Once the unit test has been created, existing code is run against it. The test should fail because the new application code hasn't been written yet.

The aim at this stage is to simply write the minimum code required to pass the unit test, and no more. When this has been achieved, the new code is run against the new test and all previous tests. The goal is to ensure that not only does the code pass the new test, but that it also doesn't adversely impact any existing features.

If all tests are passed the project moves to the final stage — peer review — before all tests are run again. Only when this has been completed is the code passed to quality assurance or pre-production.



Based on SOLID programming

Many organizations struggle to implement automated unit testing and test-driven development effectively. This is often because the traditional design and development processes they use don't deliver the required outcomes.

An object-oriented methodology will ensure that applications are easier to maintain and extend over time but there are some key concepts that need to be considered when adopting this type of approach.

In our experience at Basis Technologies, successful unit testing and test-driven development programs are based on the five core principles of SOLID. In particular, principles 1 and 5 — Segregation of Responsibility and Dependency Injection — are crucially important.

S	O	L	I	D
Segregation of Responsibility	**Open Closed Principle**	**Liskov Substitution Principle**	**Interface Segregation**	**Dependency Injection**

5 core principles of SOLID

1 **Segregation of Responsibility**
The new code being developed should only do one thing

2 **Open Closed Principle**
Classes should be open for extension and closed for modification. Instead of modifying a class that does not quite fit the needs, a new class that inherits the features of the original class and adds new features should be declared

3 **Liskov Substitution Principle**
Derived classes must be completely substitutable for their base classes. Although base class functionality can be overridden it should only be done with great care, as it can create confusion

4 **Interface Segregation**
Classes should not be forced to rely on interfaces that they don't need. The fact is that it's better to have many specific interfaces than it is to have one large interface.

5 **Dependency Injection**
A strategy for decoupling software and simplifying unit tests. Creating mock dependencies that mimic the behavior of real objects in controlled ways enables all the external dependencies of a software unit under test to be controlled.

5 steps to a successful culture shift

The concepts discussed here can be difficult for programmers steeped in traditional software creation techniques to assimilate, as they apparently turn the whole process on its head. Yet getting developer buy-in is essential to making automated unit testing and test-driven development work, and benefiting from the resultant gains. Consequently, it is essential to put a robust change management program in place.

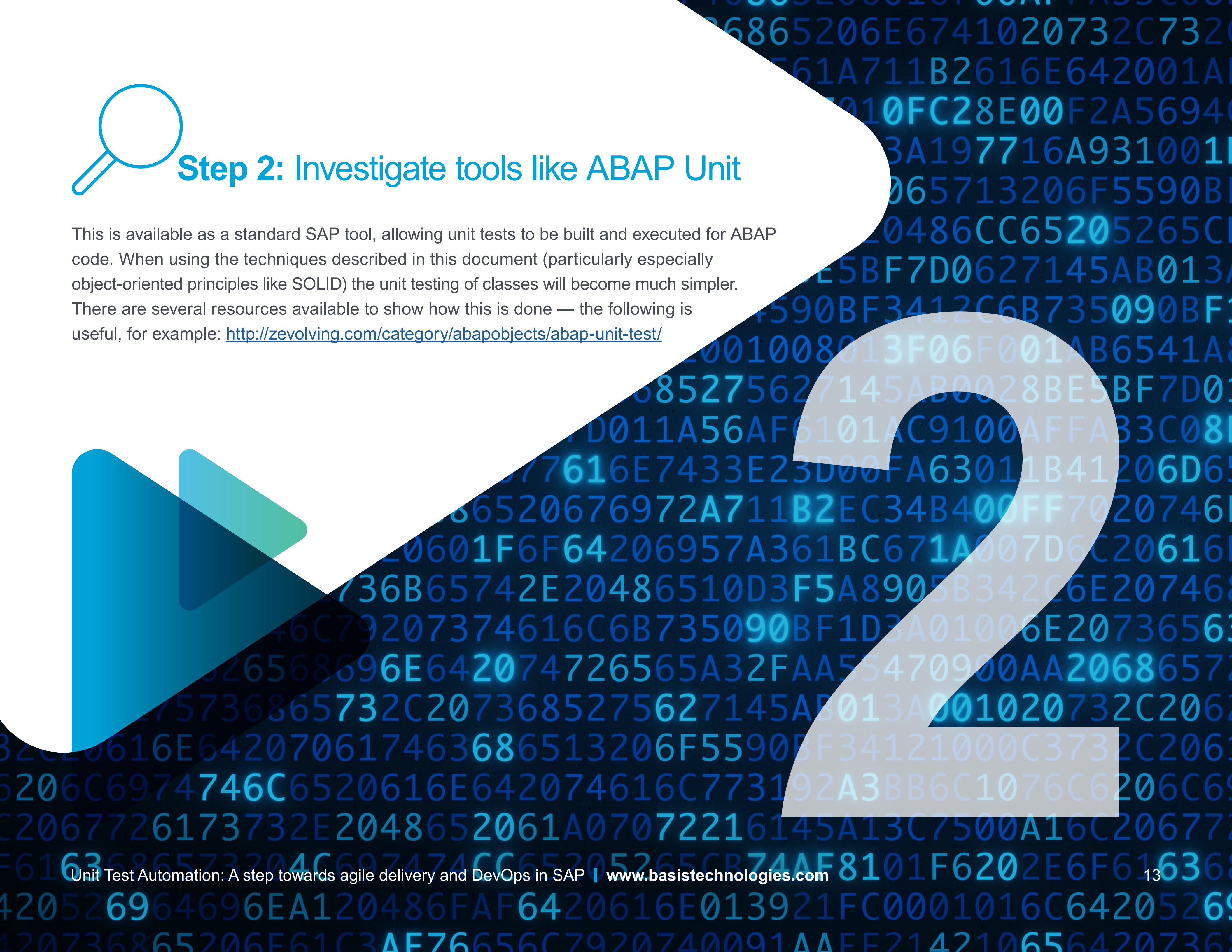
Following these steps will ensure a smooth transition…

Step 1: Enlist volunteers

Once developers are trained, particularly in the principles of SOLID, and start to see how effective the process is, they will become advocates and win over their less enthusiastic colleagues. In our experience, once programmers start using these concepts they become more committed and have a greater sense of ownership about the code they create. Their work becomes more enjoyable and provides greater value.

Step 2: Investigate tools like ABAP Unit

This is available as a standard SAP tool, allowing unit tests to be built and executed for ABAP code. When using the techniques described in this document (particularly especially object-oriented principles like SOLID) the unit testing of classes will become much simpler. There are several resources available to show how this is done — the following is useful, for example: http://zevolving.com/category/abapobjects/abap-unit-test/



Step 3: Create a sense of urgency

Start to request and implement functional changes on a more frequent basis, rather than storing everything up for major releases. Make a decision that automated unit testing will be implemented from the very start of the next new development or project, and ensure that nothing leaves development unless unit testing has been successfully carried out.

Step 4: Establish the right environment

Establish an environment in which people can make mistakes without fear of criticism. This is a major change and it will take time for developers to adapt to the new techniques and tools. Introducing peer reviews of code and unit tests will help to share knowledge and get people up to speed.

Step 5: Point out the career-enhancing benefits

Point out the career-enhancing benefits and the positive effect on employees' CVs. As the pace of digitalization hots up, and SAP reinvents its software portfolio for the digital economy, the requirement for fast, accurate releases will only increase. As a result, developers that have the necessary skills to support agile development will be in great demand.

“Delivering faster doesn’t require organizations to cut quality corners or drive people to unsustainable overwork. It does require Application Development & Delivery leaders to **fundamentally change the way their teams organize, staff, collaborate, and automate.**”

Kurt Bittner, Forrester

Source: Forget Two-Speed IT; DevOps Enables Faster Delivery Across The Board Vision: The Modern Application Delivery Playbook by Kurt Bittner, Forrester, November 6, 2015.

Conclusion

Automated unit testing and test-driven development offer organizations the opportunity to shift-left quality control and address defect fixes at the development stage. Whether combined with a DevOps approach or even agile and more traditional project methodologies, it enables organizations to dramatically reduce the cost of software defects, accelerate the development of code, and deliver new functionality faster. But that's not all. Though this kind of cultural shift may require significant amounts of persuasion and effort up-front, the positive impact is not only seen during the development of new features.

IT organizations typically spend 70-80% of their budget on 'keeping the lights on', and improving existing products and services. Integration of automated unit testing into the development process ensures that future maintenance will become cheaper, less time consuming and a lot less risky. That means scarce IT resources can be allocated to work that can deliver more value, more quickly.

As a result, companies can adapt their systems of engagement faster to take advantage of changing consumer expectations, new technology-driven opportunities, and dynamic market, economic, and business conditions.

Unit testing…

- increases product quality
- increases agility
- improves digital transformation

The DevOps and Test Automation Platform

ActiveControl

ActiveControl allows SAP systems to respond to new business requirements at high speed. Its range of powerful automation features support agile development, DevOps and Continuous Delivery in SAP environments, generating more business value through faster, safer application delivery. It accelerates change without putting business continuity at risk, providing fast, efficient change management that puts you in complete control of your SAP change and release processes.

Testimony

Testimony is a fully automated, next-generation testing tool which uses Robotic Test Automation to create a comprehensive regression test library and eliminates the need for test script creation and maintenance. Without scripting, regular regression testing becomes a reality in the short ‘sprint’ (development) cycles typical of agile and DevOps approaches, and provides a practical way for development teams to shift testing left.

At Basis Technologies, we develop automation technology that massively reduces the time and effort needed to execute SAP change and testing. We are committed to driving business agility and transformation through agile development, DevOps and continuous delivery.

Our software — the only complete DevOps and testing platform engineered specifically for SAP — enables companies to accelerate innovation, ensure continuous quality and delivery, and lower risk across even the most complex SAP landscapes.

Since 1997, we have helped enterprises become more agile, innovative, and competitive. Our platform is SAP Certified for use on both S/4HANA and ECC systems, which is why many of the world's leading companies trust our subscription software to help them succeed in the digital age.

Contact us to find out how we can help your business adopt DevOps for SAP

www.basistechnologies.com

basis