|  |
| --- |
| Austin community college |
| Software Testing’s Monetary Ups and Downs |
| ITSE 1391 Introduction to Software Testing |
|  |
| **Christofer Jones** |
| **11/26/2018** |

|  |
| --- |
|  |

Table of Contents

[Why is Testing Necessary? 2](#_Toc532474347)

[Static Testing 2](#_Toc532474348)

[What is Static Testing? 2](#_Toc532474349)

[Advantages of Static Testing 2](#_Toc532474350)

[Disadvantages of Static Testing 3](#_Toc532474351)

[Dynamic Testing 4](#_Toc532474352)

[What is Dynamic Testing? 4](#_Toc532474353)

[Advantages of Dynamic Testing 4](#_Toc532474354)

[Disadvantages of Dynamic Testing 4](#_Toc532474355)

[Static Versus Dynamic 5](#_Toc532474356)

[Costs of Testing 5](#_Toc532474357)

[Cost of Static Testing 5](#_Toc532474358)

[Cost of Dynamic Testing 5](#_Toc532474359)

[When Testing is not Done… 5](#_Toc532474360)

[Areas Most Affected 6](#_Toc532474361)

[Failures Vary by Industry 6](#_Toc532474362)

[Monetary Cost of Software Bug Fixes 7](#_Toc532474363)

[Time the Bug is Found 7](#_Toc532474364)

[Ways to Reduce Testing Costs 10](#_Toc532474365)

[Timing 10](#_Toc532474366)

[Test Strategy/ Method 10](#_Toc532474367)

[Return of Investment (ROI) 11](#_Toc532474368)

[What is the Automated Testing ROI? 11](#_Toc532474369)

[Conclusion 12](#_Toc532474370)

[Bibliography 13](#_Toc532474371)

# Why is Testing Necessary?

Testing helps to “determine the quality and performance of the application” (testbytes). Without testing, most products, if not all, will have never made it long. Since testing helps show the quality and in turn improve quality of products, anyone can agree that testing holds an important role in the everyday life of mankind and technology’s interaction. Every little application update notification that appears on one’s cell phone is an example of what testing does for humanity. Those ‘bug fixes’ that fix the problems that may be experienced in the application or software is all thanks to the testing process.

Everyday, millions, if not billions, of people get on a mobile device and click on an application. Little do they realize that for them to even have their cellular phone, it has been tested, hardware and software, to meet the quality needs and standards of the population.There are some types of testing that directly affect you and you may not be able to know it does. The types to be discussed are: static testing and dynamic testing. The next few sections will discuss some details about each of the testing types. What are some types of test? How do they affect the human-technology interaction? Let’s examine testing, starting with static testing.

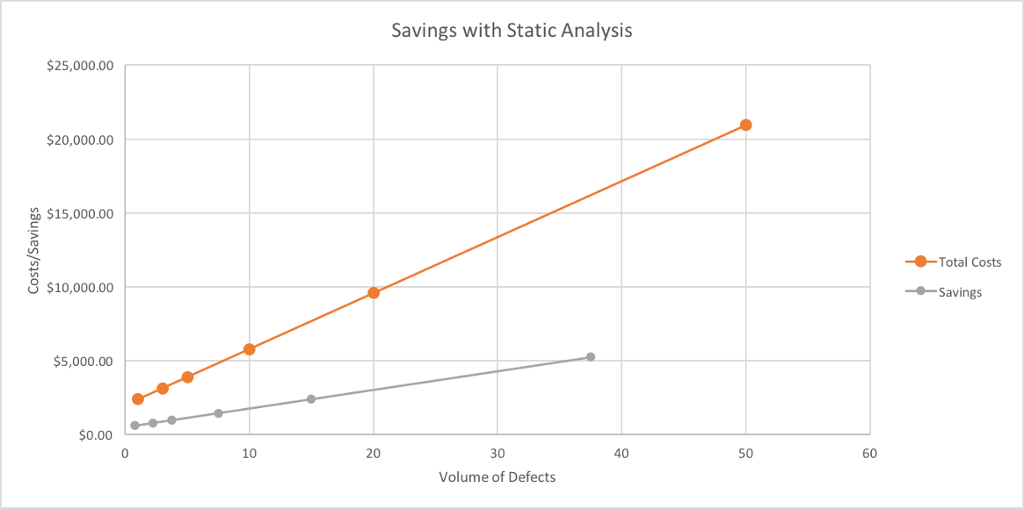
# Static Testing

## What is Static Testing?

According to one reference work, “Static testing is any evaluation of the software or related work products (such as requirements specifications or user stories) that occurs without executing the software itself” (Black 8). This means that static testing does not execute the software, application, or code. Rather, the code or item is analyzed in order to evaluate pieces of software. Since it occurs early in the developmental cycle, “it is also referred to as verification testing” (testbytes).

## Advantages of Static Testing

There are many advantages of static testing. One of which is that it helps to identify coding flaws such as logic errors. This is important because some dynamic test cases won’t expose a logic error. The code must be analyzed for this purpose. This method of testing is also advantageous because it reduces the cost of fixing bugs due to its being at the beginning of the developmental process. This reduces the number of bugs that are introduced to the software later in the process, thereby reducing lost time and costs of late development fixes.



(Courtesy of: (Graham))

## Disadvantages of Static Testing

Though there are many advantages of this test method, there are always disadvantages. One that exists is that the tools used for this method of testing only work in a certain limited amount of programming languages. The problem with this is that makes the tester must either learn the programming languages or search for automation tools in the languages they know well enough to automate the tests.

Another disadvantage is the fact the amount of time this method takes when done by hand. If automation is not used, it “takes 20 to 25 days to learn” static testing manually (Gupta). That is just to learn it not to mention the time it may take to execute depending on the software being examined and tested. Now, it is true that automation cannot do all manual testing does, so it is necessary to have manual testing still. This is just in the case of how long it takes to learn and if the automation doesn’t exist. Manual testing is a time intensive activity.

Yet another disadvantage is the provision of “false positives and false negatives”. These can be really aggravating considering a false positive says the case passed when it failed, and a false negative says a case failed when it passed. This also feeds into the amount of time it takes to use this method of testing. In any business, time is money. It may seem a good thing to take a ton of time to test something, but in reality taking an excessive amount of time testing wastes time for other stages in development, which is a loss of revenue.

# Dynamic Testing

## What is Dynamic Testing?

Dynamic testing is just the opposite of static testing. “Dynamic testing is an evaluation of that software or related work products that does involve executing the software”, meaning there must be an execution of a product not just an examination based upon code. This execution can include things such as checking reports made by the application for accuracy or response time the application has with user input.

## Advantages of Dynamic Testing

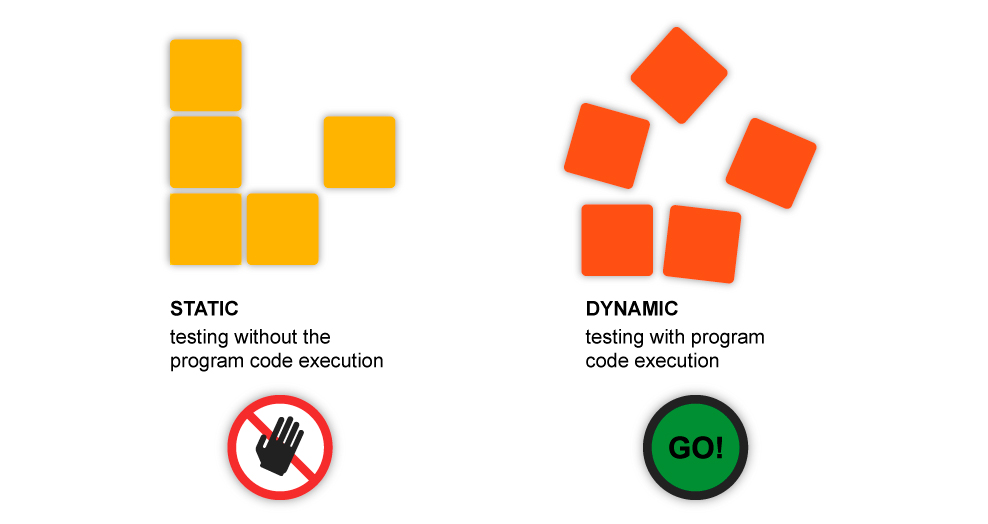
There are a few advantages of dynamic testing. One of these is it helps in “identifying weak areas in a run-time environment” (testbytes). The run-time environment is exactly what it says; it is the environment that an application runs in. A weakness in this area would mean that while testing the application in the environment set up, a weakness in the software is revealed while running, thus revealing anything that could present itself as a problem while the users run the application. It is obvious how this can help. Another advantage of dynamic testing is that finds weak areas that may not be seen with static testing. In turn it validates the static test findings.

One very big advantage of dynamic testing is that it can be applied to any application. Every application is run somehow, so it would be logical to test how it runs and functions before its release, would it not? This also makes this an advantage because it covers a large spectrum of languages instead of a limited amount, thus forcing the tester to learn a new one. This also feeds into an advantage of being able to analyze an application without having its actual source code to analyze. An example of this would be beta testing a video game. In this case, a gamer can test the run and execution of a video game without having its source code.

## Disadvantages of Dynamic Testing

As with most things in life there are disadvantages. One major disadvantage is that the ability to “trace the vulnerability in the code” is difficult, “it takes longer to fix the problem. Thus, it becomes costly to fix errors” (testbytes). When it is difficult to trace a weakness in software, it makes it that much more expensive in the long run because bugs and defects will present themselves later in the development process, thus costing a company more. Another disadvantage that feeds into this is the fact that it is difficult to find professionals who are trained in the art of dynamic testing.

### Static Versus Dynamic



(Courtesy of (MustKnow))

# Costs of Testing

Now that some examples of two frequent types of tests were discussed, it is important to know what the cost of these tests are. In this section, the costs and losses of revenue will be discussed, along with a few statistics and cost analysis graphs and figures to better explain.

## Cost of Static Testing

The cost of static testing, the finding and fixing defects, will be lower in the long run. Why? Well since static testing is code examination, which happens early in the SLDC (Software Development Life Cycle), it costs less to do these tests. The ROI (Return on Investment) is high also because of this reasoning and its placement in the life cycle of development.

## Cost of Dynamic Testing

Dynamic testing, the “cost of finding and fixing defects is high” (Guru99 Tech). Since testing the actual software comes after the software development stages, it is going to cost more to fix. This also means that the “[r]eturn on investment will be low as this process [is] involve[d] after the development phase” (Guru99 Tech).

## When Testing is not Done…

Software that is left untested can be dangerous not only to the environment, but also to mankind and its safety. All the machinery that saves lives, for example, precision surgery robots, laser eye surgery machines, all these machines could cause death if not tested time and again by testers and software companies. Many other drastic things can happen due to instability in software. Privacy leaks and lost account information can happen due to lack of security checks. Money can be taken from users without their knowing due to the same lacking area. There are less drastic things that could happen, inconveniences. These include lack of good interaction or visual quality, no results, or even the program may lack the ability to function completely.

Lack of testing, or even starting software test early in the developmental process may also cost tons of money. As time goes on, it costs more and more revenue in order to remove the software defects.

“In 2002, software bugs cost the United States economy approximately $59.5 billion. In 2016, that number jumped to $1.1 trillion There are the more obvious costs such as revenue lost due to customers being able to use the product and payments to the developers who must fix the bug. However, there are less obvious ones. When a software failure is made public, companies lose an estimated $2.3 billion in shareholder value on the first day alone. There are indirect financial costs coming from problems with brand reputation and customer loyalty. Fixing the bug can also push back other code changes and new features as well as mess up the production schedule.” (Cohane)

# Areas Most Affected

Different sections of the economy and the world’s infrastructure are affected by software failures. In 2017, $1.7 Trillion (an estimated: $1,715,430,778,504) was lost due to failures in software. What was affected? Well, a testing company named “Tricentis found that retail and consumer technology were the areas most affected, while software failures in public service and healthcare were down from [2016]” (Matteson). Why is this? Well businesses and their patrons rely on software for the variety of services offered and the functional possibilities offered by the accompanying technology.

“In a recent report, software testing company Tricentis analyzed 606 software fails from 314 companies to better understand the business and financial impact of software failures. The report revealed that these software failures affected 3.6 billion people and caused $1.7 trillion in financial losses and a cumulative total of 268 years of downtime” (Matteson).

## Failures Vary by Industry

Different industries will have different levels of affectedness. For instance, as previously mentioned, the retail and consumer technology industries reported the most failures and problems, mostly due to the “problematic smartphone updates and the security/hacking exploits intended to target them” (Matteson). Public services and health care saw failures lower 30%, but still having a few issues. These include such things as “hacking and tampering with international election processes, WiFi vulnerabilities that exposed the data of billions of people, IT issues that caused tens of thousands of letters that went unsent to patients and doctors, millions of dollars in overpaid medical billing”, and so on and so forth (Matteson).

## Monetary Cost of Software Bug Fixes

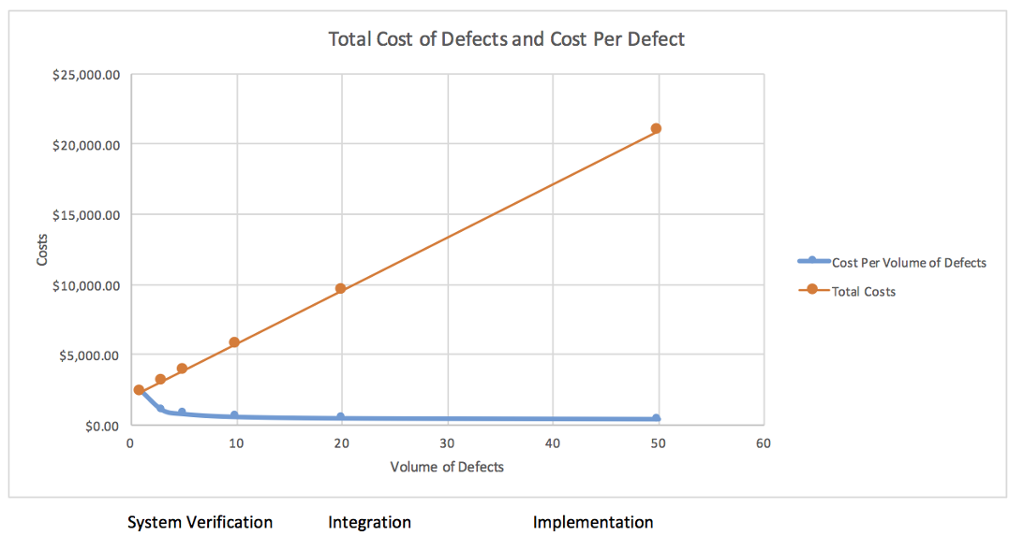
The cost of software bugs may vary depending on a few factors such as:

* The Severity of the Bug
* Type of Bug
* How late or early the bug is found

The main one to be considered here is how late or early the bug is found.

### Time the Bug is Found

Timing is everything. Perhaps you have heard this before. Well, in the world of software and technology, this statement is true. Especially, in the aspect of bug detection and resolution. “The cost of a bug goes up based on how far down the SDLC (Software Development Life Cycle) the bug is found” (Leon, The True Cost Of A Software Bug: Part One). What does this mean? Well, to fix a bug early in development, it is rather simple. The product is not released yet and there is no wasted or lost time, so the bug can be fixed in the appropriate part of the development process.



(Courtesy of: (Graham))

When the bug is found later in the development process, it costs a company more to fix their product. When a bug is found,

“[T]he code needs to go back to the beginning of the SDLC so the agile development cycle can restart. Then there’s the domino effect to think about. The software development approach often need to change to accommodate the code fix, which can in turn bump back other code changes. SO no only is the bug going to cost more to fix as it moves through the second round of SDLC, but a different code change could be delayed, which adds cost to that code change as well” (Leon, The True Cost Of A Software Bug: Part One).

Here is a pictorial representation of estimated costs:



(Courtesy of instinctools.com)

As we can see, this is proved true. The further in the process the bug is found, the more the cost is per bug to be able to follow the resolution process. How is this calculated? Or, why does it work like this? Well there are some implications of each stage. These stages are:

* Requirements
* Coding
* Code Integration
* Testing
* User Acceptance testing
* Production

#### Requirements Stage

The way the cost is calculated in the requirements phase of developments is rather simple. The only thing that can be charged and that costs is “the time it takes to rewrite the requirement” (Leon, Bug Cost Implications At Each Phase Of The SDLC: Part Two). Now, is it a little clearer as to why it is best to start testing early? The amount of money it costs to rewrite a few requirements is very minimal. It is only charged based upon one thing.

#### Coding Stage

Things are a little different in the coding stages of the SDLC. The cost starts to include “additional required developer hours” which costs more than just rewriting the code requirements (Leon, Bug Cost Implications At Each Phase Of The SDLC: Part Two). There are more people to pay in this stage. The time they are paid for may “var[y] based on complexity, but is less than fixing a bug found by someone else” because at least when a developer comes across a bug in their own code, “they already understand the problem and know how to fix it” (Leon, Bug Cost Implications At Each Phase Of The SDLC: Part Two). The time it would take, and the cost to fix a bug in another person’s code would be dramatically different.

#### Code Integration

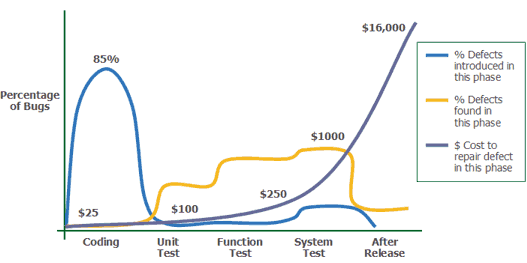
In this stage of development, the cost is twice as much as the precious stage of development. Here, since it is a later stage and errors may occur at a higher level, there are more people to pay, and more time is also involved since the exact piece of code with an error needs to be found, understood, then reworked. This can push time back and product reveal time back.

#### User Acceptance Testing

User Acceptance Testing is usually when the customer makes the producing company aware of a flaw or something isn’t working correctly. A more common term for UAT is Beta testing. This is the process of sending the user a piece of software in order to test that the software product is in normal working order. If the tester of the software “can reproduce the bug then he or she must document the steps, submit a defect, prioritize the defect, and meet with developers to discuss” (Leon, Bug Cost Implications At Each Phase Of The SDLC: Part Two). As can be imagined, this may cost quite a bit considering this is the last phase of development before finalization and production. The cost is the expenses and time to use developers, system engineers, project managers, the customers time, and quality assurance time afterwards. A hefty cost.

#### Production

During production, a phone call is placed to technical support and then based on the complaint it is decided on whether there is a bug or if the product is working properly. Then, “[t]he customer is notified, the PM [(Project Manager)] is contacted, and then the process in the UAT [(User Acceptance Testing)] stage is followed”, so there are more costs due to the prominence of the employees being paid and the amount of QA, or Quality Assurance hours (Leon, Bug Cost Implications At Each Phase Of The SDLC: Part Two).



(Courtesy of: (Thief))

# Ways to Reduce Testing Costs

There is always a solution to a problem, whether it is identified or conceptualized or not. In the realm of software testing, some things can be fixed with timing, others are strategically based.

## Timing

As earlier mentioned, a software bug is “4-5 times cheaper to fix when caught in the early stages of development compared to post release fixes” (Screenster). So, what would be a good solution to the problem? Well, starting testing early, and more testing early on would be a good idea. The amount of time and work that goes into finding, naming, fixing, and retesting defects differs across all units. But, if tests are run earlier in the SDLC, there will be a greater coverage of the possible bugs in the system.

## Test Strategy/ Method

Sometimes the way something is done may be an ultimate downfall. One way to prevent the huge costs of bug removal is to “minimize manual testing from day one” (Screenster). Manual testing may seem like a burden to some testers. This is due to the long amounts of time and effort spent doing things manually. This is where the automated testing comes in. By reducing the use of manual testing and replacing it with automated testing, the cost of testing is reduced greatly. Time is money, as the expression goes. The less time spent on these manual tests, is less money spent on testing.

Another method is to “”[d]etermine ‘stable’ and ‘well tested’ areas of the application to minimize testing in these areas” (XBOSOFT). Why? Well, there may be areas of testing that need a real scrutinous eye and a thorough manual test. If one were to manually test every component of the software project a lot of time would be wasted, as this isn’t necessary. In turn, a ton of money is wasted. So, by taking a load off by using automation for the areas that need less testing, more time can be spent in more needed areas of the project.

Another way, which works with most things in life, not exclusively software, is the ability to remove and reduce the amount of unnecessary and expensive items. In terms of software this means the ability to know what isn’t useful to the project and to know what is costing too much. It is beneficial to find “open source or less expensive options and remove the rarely used commercial tools which bring minuscule benefits” (Ness).Not only can the cost of tools be reduced, but also, the maintainability and usability can be reduced. By choosing a few tools that are easy to understand, and use, the cost of time to learn the tools and maintain the standard becomes less. Here are a few of those tools for future reference:

* Selenium (Info By: (Selenium HQ), (Software Testing Help))
  + Learning a scripting language not required
  + Runs in many browsers and operating systems
  + Can be controlled by many programming languages and testing frameworks
  + Compatible with several programming languages and automation
  + Can create scripts…that is of great help for prompt reproduction of bugs, regression testing, and exploratory testing
* TestComplete By SmartBear (Info By: (SmartBear Software), (Software Testing Help))
  + Support for 7 different scripting languages
  + The ability to record robust automation tests without scripting knowledge
  + Comprehensive defect tracking and reporting capabilities
  + Can scale your tests across 1500+ real test environments for complete coverage and improved software quality using TestComplete
* Silk Test By Microfocus (Info by: (Software Testing Help))
  + Aims at automated functional and regression testing
  + Cross-browser support and provides unified test automation for a variety of applications including [desktop, mobile, web, rich-client] and enterprise applications
  + Enables efficient, speedy and high-quality automation testing

# Return of Investment (ROI)

What is the ROI or Return on Investment? Well, it is defined as “a widely used measure to compare the effectiveness of IT [(Information Technology)] systems investments. It is commonly used to justify projects but can measure projects returns at any stage” (Axia Consulting).

## What is the Automated Testing ROI?

Let’s start with a misconception: Automation is free. Why is this not true? Well, the big thing people forget to consider is the cost.

“It requires tools – and specialized skills to use them. It takes time for manual testers to provide sufficiently detailed documentation to guide the automation engineers; for the engineers to develop, test, and maintain tests; and for manual testers to verify that the tests are executing as expected and to analyze and unexpected results” (Hayes).

There really is no such thing as “free” when it comes to testing software. Everything costs, and this is no exaggeration. Everything from time spent testing a piece of software, to the tools used, to the payout given to the employees. It all costs. Now to be able to reduce this cost, and have a higher ROI (Return Of Investment), what can be done? Well the way to “save money [is to] test *more*” (Hayes). To deliver a quality product the first time is a matter of doing as much testing as possible, with the help of some automation where efficient to do so, and not have to pull the product back in later in development or even after development. This will result in a higher ROI.

# Conclusion

It turns out that software testing is very important to the economy. Without it there are major damages such as consumer loss of funds, a business not being able to contact its patrons due to hacks or security failures. We saw this recently with the bank hacks and people losing their money. We also learned that there is no such thing as testing too much and testing too early. This can prevent the above-mentioned damages and disasters. I learned how important my future role as a tested is. Without more people to do this job, the world and the security of entire nations could be at risk if things are not patched and bugs and defects are not found. There is more than monetary ups and downs but national and global downs if something is not fixed correctly.

# Bibliography

Axia Consulting. *Basic ROI Calculation*. 2018. Webpage. 12 December 2018.

Black, Rex. *Foundations of Software Testing*. Cengage Learning EMEA, 2012. Print.

Cohane, Ryan. *Financial Cost of Software Bugs*. 15 November 2017. Webpage. 28 November 2018.

Graham, Bill. *The ROI of Static Analysis in Safety-Critical Software Development*. 2 December 2015. Webpage. 13 December 2018.

Gupta, Pravin Kumar. *Quora: How long does manual testing course takes to finish?* 3 June 2018. Blog Post. 29 November 2018.

Guru99 Tech. *Static Testing Vs Dynamic Testing*. 1 December 2018. Webpage. 7 December 2018.

Hayes, Linda. *What is the ROI of Test Automation?* 4 November 2015. Webpage. 13 December 2018.

Instinctools. *Why QA Is So Important For Software Project*. n.d. Webpage. 4 December 2018.

Leon, Janet. *Bug Cost Implications At Each Phase Of The SDLC: Part Two*. 4 March 2014. Webpage. 6 December 2018.

—. *The True Cost Of A Software Bug: Part One*. 28 February 2015. Webpage. 4 December 2018.

Matteson, Scott. "Software Failure Caused $1.7 Trillion in Financial Losses in 2017." 26 January 2018. *TechReupublic.* Webpage. 13 December 2018.

MustKnow. *Manual, Static & Dynamic Testing*. 2 April 2018. Webpage. 7 December 2018.

Ness. *Optimize Testing and Reduce Software Testing Costs*. 9 December 2015. Webpage. 10 December 2018.

Screenster. *5 Strategies to Reduce Software Testing Costs*. 30 January 2018. Blog. 8 December 2018.

Selenium HQ. *What is Selenium?* n.d. Webpage. 12 December 2018.

SmartBear Software. *TestComplete Features*. n.d. Webpage. 12 December 2018.

Software Testing Help. *Top 20 Best Test Automation Testing Tools in 2018 (Comprehensive List)*. 8 November 2018. Webpage. 12 December 2018.

testbytes. *Pros and Cons of Dynamic Testing and Static Testing*. 13 July 2018. Webpage. 28 November 2018.

Thief, Atlantis. *Cost of bugfixing*. 2015 February 2015. Electronic. 6 December 2018.

XBOSOFT. *10 Ways to Reduce the Cost of Software Testing-Agile Testing*. 19 March 2013. Webpage. 10 December 2018.