What techniques can be used to know when to stop testing?

Anne Line Gjersem, Jenny Martinsson, Kristoffer Mårtensson, Mergim Rruka
Dept. of Comp. Science, Lund University
{dic14agj, dic15jma, fpr15kma, me7076rr-s}@student.lu.se

***Abstract*—This paper presents and analyzes five different software stop testing techniques for the purpose of deeper education within a chosen area of software testing. To achieve this purpose, two companies were made up which were in need of a stop testing technique, a small sized startup company and a big sized airline company. The advantages and drawbacks for each technique were discussed, followed by an analysis of how well each technique suited the needs of the two companies. Three of the techniques were well suited for both companies whereas the two others were suitable for the small startup company.**

I. Introduction

Releasing a software as soon as possible is desired by many companies. Early releases can give the benefit of being introduced to the market, to achieve this the software needs to be tested. The number of possible paths can be very large in a software program therefore it is not feasible to find every possible bug [1]. If the software is not tested enough it can mean that it can contains bugs, poor quality code and therefore hurt the company in the long run. Some software's might not be sensitive to containing bugs when released while other might lead to fatal accidents if they do.

A software that is going to be part of an aircraft is going to need a lot of testing and fixing before it can be used in an actual aircraft and in an actual flight. The software might contain thousands of lines that control everything from the lights in the aircraft, to the control systems. The aircraft companies want to ensure that everything works because there is no room for errors in cases like these where a big amount of human lives are involved [2].

This is not necessarily the same case when it comes to a website. A website for a startup that is about social media might not have to make sure that everything their software does is done faultless. In this case, there might be more room for error than compared to an aircraft software.

A. Scope

The aim of this report is to identify stop testing techniques and evaluate if the stop testing technique is applicable to a given system. To limit the research area two scenarios has been set up.

Company A: 'start up company'
The first scenario is a start up company that uses agile methods for development software. The focus is on continues delivery and working in sprints. The company delivers social media web applications for their customers. The system quality needs to be 'good enough' to create value for the customers. The financial resources is limited.

Company B: 'Aircraft system'
The second scenario is a company that develops software for aircraft system. The software is classified as critical since an failure can cause a lot of damages. Even the slightest bug can be vital and have a negative impact. Hence, the software has to be of high quality.

This report will address the following issues:

1) What techniques can be used to know when to stop testing?
2) What are the stop testing techniques advantages and drawbacks?
3) Are the addressed techniques suitable to use in company A or company B ?

II. Description

There are different frameworks and techniques on when to stop testing a software, in this section some of these techniques and frameworks will be explained.

A. Software Reliability Growth Models

Reliability Modelling is used to predict the reliability of the system in the future [3]. This is then used to estimate a reliability level that determines how much testing is needed before the system is stable for release [4]. Software Reliability Modelling is made up of several different techniques, that all have their pros and cons, and all can be used to estimate when to stop testing [4].

Software Reliability Growth models (SRGMs) are techniques used to describe software failure and fault-detection in the system [4]. The goal of these methods are to try to trade off between quality, schedule and cost during the testing and validation phase [4]. SRGMs address the mathematical relationship between different attributes that affects the process of testing [5]. SRGMs take failure data from an earlier period when faults were being detected and corrected, either during testing or during operational use, to estimate how the reliability will change in the future [3].

1) Time between failure: There are several different Software Reliability Growth Models, divided in different types. One very common method is looking at the time between failures. There are several different models within this type, but one model is the Jelinski-Moranda method. This method evaluates the reliability of a system by keeping track of how often it fails. If the time between each failure is increasing, we know that the system is becoming more reliable [6].

2) Failure Count: One way to test reliability is by using failure count. The most commonly used technique using failure count is the Goel-Okumoto model, also known as Non-homogeneous Poisson Process [7]. It's very useful approach to determine for how long the software needs to be tested [8], and therefore when to stop testing.

B. Value Based Approach

With a value based approach the test cases are derived from risk analysis. The risk exposure is calculated by multiplying loss probability and loss impact. To decide which test cases to prioritize, the prioritized requirements are considered by the project decision-maker [9]. One stop testing technique that can be used in value based a approach is the combination of the models: constructive cost model (COCOMO II), constructive quality model (COQUALMO) and value-estimating relationships (VERs). The stop test method can be used to determine what quality level is enough for different situations [9].

1) Constructive Cost Model -COCOMO II : When developing a software product cost estimation COCOMO II model can be used to estimate the required effort [9]. The effort is measured in person-month and the estimation of the project size is measured in thousands lines of code (KLOC). Furthermore, the model considered personnel attributes such as experience and capabilities, project characteristics such as execution time and storage constrains and product characteristics such as complexity and required reliability [9]. The trade off between cost, reliability and test time can be visualised using the COCOMO model in a regression analysis. The required reliability is referred to as RELY. If the value of RELY is high it means that failure of impact is critical, and a low RELY the impact is considered to be inconvenient.

2) constructive quality mode-COQUALMO: Delivered defect density can be estimated with the COQUALMO model which is an extension off the COCOMO II model [9]. In COQUALMO there are two submodels: defect introduction model and defect removal model. Defect introduction model estimates defects occurring requirements process phase, design process phase and code process phase. The defect removal model considered defect which occurs i the following activities: automated analysis, people reviews and execution testing and tools [9]. A rating scale from very low to extra high is used to determine what method and techniques to use for peer reviews, how much to investment in automated analysis tools and how to execute the tests. "Very low" indicates that now testing tool or peer review is necessary."Extra high" indicates that peer reviews should include formal review with roles and procedures, root cause analysis, detailed checklist and the tools used are highly advanced [10].

3) Value-estimating relationships (VERs): When performing value based testing, it is crucial that critical stakeholders of the project provide value-estimation relationships (VERs) to the developers. VERs show the relationship between quality levels or delivery times for different parts of the software and the benefit flow or values earned from that part of software. The idea is to get a better understanding of which parts of the system deliver the most value in order to know where the most effort should be put [9].

C. Value-Based Software Quality Model

The value based software quality model(VBSQM) includes risk analyses using the RELY rating for a business case, a size of loss (Sq(L) rating) for the system and the project size is measured with the model COCOMO II [9]. The VBSQM combines risk exposure and can find the sweet spot for the software quality investment level [9].

D. Exit Criteria

Another approach to know when to stop testing is exit criteria. Different criteria can be predetermined for each new test cycle that will begin. This means that the developing team and testing team know what to look for when deciding if testing cycle is finished or not and in this way decide what to the next step is. This is approach is an addition to the testing method that the testing team uses, meaning that the team decides what technique to use when testing and this approach helps to understand when the testing should end [11].

There are different criteria that can be considered depending on the test cycle. These are a few that should be considered: [12]

- Major defects are identified and solved.
- Critical test cases are passed.
- Functional coverage is fully completed.
- Test coverage has reached a certain percentage.
- A certain percentage of test cases are allowed to fail if they are of low priority.
- Time and budged are depleted.

An outcome of these criteria can be: [12]

- Summary of findings.
- Logs of the tests.
- Logs of reported test incidents.

E. Post release testing

A different approach to the issue of testing versus release of software is the post release testing approach. This approach is very beneficial since it allows the software to be introduced to the market in time or even before the planned release. Of course, there is still a need for the software to be almost fully functional but it leaves open doors for smaller bugs to be detected after the software's release. The testing continues after the release and combined with bug reports from the users, it makes it possible for the team to release patches afterwards. These patches can address bugs that were found by the post release testing or address the reported bugs from the user [13].

Figure 1. shows how the usual approach of testing and release works. The testing cycle has a start and stop time that does not overlap with the product after it is released. In this case all the tests that needed to be done, had to be done before the release [13].

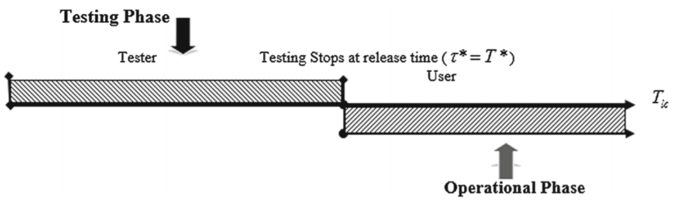

Fig. 1. Usual approach of testing and release [13].

Figure 2 shows how the post release testing approach works. The testing phase is divided in two parts, the pre-release and post-release phases. The post-release phase is the testing phase that continues the testing after the software is released [13].

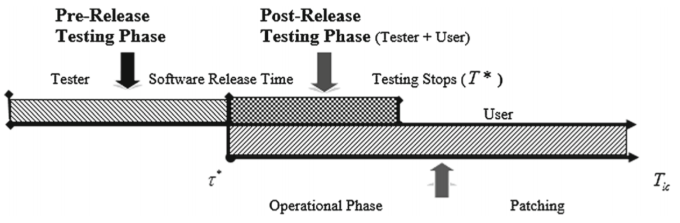

Fig. 2. The post release testing approach [13].

F. Capture Recapture

Capture recapture is originally a technique used to track population of wildlife. This is done by capturing animals and marking them and continuously keeping track of when animals are recaptured. With the data stating how many animals were recaptured how many times, an estimation of the population can be made. In software testing, this can be used by allowing more than one inspector search for defects in a system and keeping track of how many defects were found or ”recaptured” by how many inspectors. By looking at the overlap of the found defects, an estimation can be made to how many defects remain in the system [14].

III. Analysis

This section will present aforementioned stop testing techniques and discuss the advantages and the draw-backs of the different techniques. Furthermore, a an evaluation will be performed to address if the techniques are suitable for system in company A or company B.

A. A value based approach

When comparing value based approach with value-neutral testing, the test cases with value based approach creates more business value per dollar invested [9]. A value- neutral testing technique is for example path testing automated test generators or test cases which is not derived from prioritized requirements. The value based approach foundation is risk analysis. Hence, one drawback of the risk exposure method is the difficulty of to quantify the probability of loss [9].

For company A ’start up company’ risk analysis can be used to determine when is the testing is good enough. By combining COCOMO II, COQUALMO and VERs in an analysis, the investments levels and the relevant software quality can be determined. Hence, it can determine what quality level is enough, when comparing the risk exposure investment with market share erosion. The advantages found for value based approach is related to business value and is therefore suitable to use for company A: ’start up’.

For company B, the RELY value is high since an aircraft system error impact can potentially lead to loss of human life and can estimate rough how much effort is going to be needed in delivering the software. In addition using the COQUALMO defect-removal-investment rating scales can indicate what level is need for testing tools, peer reviews and automated analysis [10]. VBSQM is suitable to use for both company ’A’ and company ’B’ to determine the sweet spot for software quality investment.

B. Software Reliability Growth Models

Software Reliability Growth Models (SRGMs) are based on previously collected data. For the SRGMs to provide anything of value then they need to be based on accurate data from previous runs [3]. However, to obtain trusted data it will cost a lot if we want to build a model that’s more than a fairly modest level of reliability [3]. One limitation is that the technique is not suited for system were testing environment is changing or new capabilities are being introduced, the failure history of the past will not be reflect when changes occurs [15].

Safety-critical systems, with high consequences of failure are not well matched with a Software Reliability Growth Model, and therefore SRGMs are not a good method to use for company B.

However, there’s also sources saying that an SRGM can be used for more safety-critical systems [16]. In this article from 2019 they researched a method called Software Failure and Reliability Assessment Tool (SFRAT) which is based on Software Reliability Growth Models. Depending on what specific method you’re using and how well you implement it, SRGM’s can be used for safety-critical systems, like company B. However it’s important to look at the data requirements to give an accurate assessment.

For company A it’s not very straight forward if it’s good to use an SRGM or not. Because of it being an agile company things will get updated and released continuously. Because of this a lot of data will be gathered, which is great for SRGMs [7]. Because of this an SRGM can be very useful, especially if it’s not in a completely new territory, seeing as you can then use data from other companies to build your model. If company A however has a very new and experimental industry, it will take some time before SRGMs would actually give any benefit.

C. Capture Recapture

Capture Recapture is a stop testing technique that have been applied and evaluated in the industry. It has been shown that it can provide relatively accurate estimations under the right circumstances and with the correct methodology. A study was made using simulations on software artifacts where they were looking to determine the impact that the number of inspectors and remaining defects in the software had on the results of the various capture recapture methods [14]. What they determined from the study was that a good number of inspectors for these methods were four. Less than four and the results were not reliable enough and more than four did not give a big enough increase in the overlap of the found defects. In regards to the number of defects in the system, it was harder to see a correlation of the success of the method and the number of defects in the software. They did however conclude that at least one capture recapture method showed a lot of improvement when looking at software with 12 instead of 6 defects.

Setting four inspectors to the task of inspecting software and searching for defects through testing can quickly become a fairly expensive ordeal. This of course is dependant on how large the code base is. For company A, this could be a viable stop testing technique. If the company has set a limit to the highest number of estimated defects allowed on a specific release, this method could provide them with this estimation. It can however as mentioned become fairly expensive due to how time consuming it would be for multiple inspectors to perform their inspection. It ultimately comes down to if the company feels they have sufficient resources and reason to perform this methodology.

For Company B this methodology is feasible but perhaps a little bit risky. The company surely have enough resources to perform this technique with possibly even more than four inspectors. The issue is however that this company can not set a limit to the highest number of estimated defects remaining, this company has to be 100% sure that there are zero defects left. This kind of eliminates the purpose of a stop testing technique which main purpose is to estimate the number of remaining defects in a system post testing.

D. Exit Criteria

The exiting criteria is something that sounds good in theory but might not always be able to be applied in practice. The criteria must be defined well and realistically since they can be the reason why the software will not fulfill them and therefore prevent the testing from being completed. Even the size of the system can affect if this method is suitable or not, for a bigger system this method might not be enough or inefficient to use. To have a criteria of 95 % coverage in a big system might be unrealistic and therefore impossible to achieve. Criteria that are loosely defined, such as "testing phase ends when all bugs are found", might only lead to problems. This type of criteria has no clear endpoint which makes it almost impossible to fulfill and end the ongoing test cycle. On the other hand, with well defined criteria, the size of the system that is being tested should not matter. This method will help teams to know when they are done and what they need to do next, since a common outcome from this method is to get a incident report and findings log.

When it comes the company A 'start up company', this method might be a good fit. Since the system is not of a bigger size and if well defined criteria are used, this method might work good. It will make it easy for the testing team to know what to look for when they are testing and when to stop the cycle. The same applies to the company B 'aircraft system', this method might work well as long as the criteria are well and realistically defined. Company B is a more sensitive system because of human lives being involved. This leads to higher requirements which might need more carefully predetermined criteria.

Depleted time and budget could be a criteria that works for the company A, however, this criteria would not work in company B. In company B the testing can not simple stop just because there is no more time or money left. If there is a risk for software to contain bugs than perhaps the testing should continue, considering that human lives are involved in this.

E. Post Release Testing

The clear benefit of this approach is the gain in introducing the software to the market in time and the ability to fix the remaining bugs after the software has been released. Moreover, it makes it possible to get free debugging which is done by the customers using the software and reporting the bugs that they encounter. However, this method still needs to be almost fully working before it is released.

This method might work well for company A, especially since it is a startup and an early introduction to the market might be a major factor in the company's success. The company can release a website that mostly works and then continue testing while the software is released. The developing team can release patches or newer version of the website after it has been launched and in this way address the bugs that they, or the users, have found. Since the website might not handle sensitive data or human lives, like company B does, than they are not sensitive to the website containing bugs. This makes it more acceptable for the website to contain bugs and therefore allow the company A to use this approach.

For company B this approach will not work at all. Since the software is for an aircraft, it is not a good approach to release a not fully working software for aircraft's and at the same time keep testing and looking for more bugs. In this case there is no room for the software to contain bugs especially since human lives can depend on it. A simple bug in the control flow system could lead to devastating events. For the aircraft company it is important that everything works as it should and it is not possible to release a not fully working software and afterwards release patches to address bugs that emerge. The only way this approach could work would be if the software for the aircraft is divided in different parts. Meaning that the parts of the software that do not pose any threats for the passengers could be approached with the post release testing method. For example, the entertainment system does not necessary need to be completely bug free and therefore it's testing could be continued after it's release. If any bugs are found, they will not risk anyone's lives and can easily be addressed with patches.

IV. Conclusion

The stop testing approaches and techniques found are: a value based approach (COCOMO, COQUALMO, VERs and VBSQM), capture recapture approach(Estimation the number of defects), Software Reliability Growth Models(including time between failure and failure count), exit criteria, and post release testing. For company A all the approaches mentioned above can work and help the company to know when to stop testing. For company B, the approaches that we see suitable are software reliability growth models, value based approach and exit criteria.

Table I. in appendix A. address the research questions by describing the stop testing techniques, listing advantages and drawbacks for each techniques and provides a description if the technique is suitable for either system at company A, company B, both of them. The last row refers to the paper were the stop testing technique is found.

V. Contribution statement

A. Anne-Line

Worked mainly on the Value Based Approach, including COCOMO II, COQUALMO and VBSQM. Both during the description and in the analysis.

B. Jenny

Worked mainly on Software Reliability Growth Models, both the description and the analysis.

C. Kristoffer

Focused on Capture Recapture and descriped and analyzed the technique. Wrote the description to VERs.

D. Mergim

Worked both on Exit Criterias and Post Release testing, both during the description and during the analysis.

E. Everyone

Discussing the different techniques, working on scope and introduction, drawing a conclusion.

References

[1] S. R. Dalal and C. L. Mallows, "When should one stop testing software?" pp. 872–879, 1986.

[2] U. M. \. Events. Getting to grips with software testing. [Online]. Available: https://www.aerospacetestinginternational.com/features/getting-to-grips-with-software-testing.html

[3] B. Littlewood, "Software reliability modelling: Achievements and limitations," *IEEE 1991*, pp. 336–344, 1991.

[4] S. J. H. M. Garg, R. Lai, "When to stop testing: a study from the perspective of software reliability models," *IET Software*, 2010.

[5] P. Kapur and A. Shrivastava, "Release and testing stop time of a software: A new insight," 2015.

[6] Z. Jelinski and B. Moranda, *Statistical Computer Performance Evaluation(W. Freiberger, ed.)*, 1972.

[7] M. X. S. N. G. L. Q.P.Hu, R.Peng, "Software reliability modelling and optimization for multi-release software development processes," 2011, pp. 1534–1538.

[8] D. R. Jeske and H. Pham, "On the maximum likelihood estimates for the goel-okumoto software reliability model," *The American Statistician*, vol. 55, no. 3, pp. 219–222, 2001. [Online]. Available: www.jstor.org/stable/2685804.

[9] B. B. L. Huang, "How much software quality investment is enough: A value-based approach," *IEEE*, 2005.

[10] A. J. B. Boehm, L. Huang and R. Madachy, "The roi of software dependability: The idave model," *IEEE*, 2004.

[11] K. Renuka. When to stop testing (exit criteria in software testing). [Online]. Available: https://www.softwaretestinghelp.com/when-to-stop-testing-exit-criteria-in-software-testing/

[12] R. Software. (2019) Entry and exit criteria in software testing life cycle. [Online]. Available: https://www.rishabhsoft.com/blog/entry-and-exit-criteria-in-software-testing

[13] P. K. Kapur, A. K. Shrivastava, and O. Singh2, "When to release and stop testing of a software," *Springer*, 2017.

[14] L. C. Briand and B. G. Freimut, "A comprehensive evaluation of capture-recapture models for estimating software defect content," 2000.

[15] W. Farr, *Software reliability modeling survey (chapter 3)*. McGraw-Hill, 1996.

[16] L. Fiondella and Y. Shi, "Software reliability and security assessment: Automation and frameworks," 2019.

Appendix

TABLE I
Table showcasing the advantages and drawbacks for each technique

Technique	**Advantages**	**Drawbacks**	**Suitable for company**	**Suitable for system**	**Papers**
Software Reliability Growth Models	Visually identify progress toward software stability. Determine the time required to achieve target reliability, time between failure, and failure intensity.	Needs a lot of data to be accurate.	A & B	Systems with a lot of data already collected.	[3], [4]
Value Based Approach (VBSQM)	A value based approach creates more business value per dollar invested.	Risk exposure method is the difficulty of to quantify the probability of loss.	A & B	Early startups, commercial and high-finance business cases.	[9], [10]
Capture Recapture	Can be used for many different types of testing. Can be scaled for different sized systems or just parts of systems.	Relatively resource demanding and only produces an estimation of remaining defects.	A	Most types of systems.	[14]
Exit Criteria	Makes it easier for the testing team to understand when the test cycle is done.	Poorly defined criteria can make the testing cycle impossible to finish.	A & B	Most types of systems as long as well defined criteria are created.	[11]
Post release testing	Allows early introduction of the software to the market.	The software must be mostly working before it's post release testing.	A	Systems where bugs do not pose any threats.	[13]