Test Optimization Through Risk-Based Validation Approach

Felix Eu
Intel IOTG Software Quality Team
felix.eu@intel.com

Tan Chin Pei
Intel IOTG Software Quality Team
chin.pei.tan@intel.com

Abstract

In real software’s project day-to-day context, there are so many validation approaches available in the market and each organization may adopt the approach that is most relevant and suitable to its business and industry. There are always the opportunities to customize the current validation process to meet the specific organization’s goal. The risk-based validation approach is useful in a situation where multiple software program releases happen and share the same validation resources but with increasing validation scope. It is intended to assist the validation team in a large organization to use the risk factors as an input, prioritize the validation tasks based on specific conditions and quality management output analysis. In contrast to the traditional software regression validation approaches which generally test everything when there is a code change, risk-based validation helps optimize a test strategy that will maintain the efficiency of the test coverage and ensure the right validation scope is implemented.

Biography

Felix Eu is a Software Quality Engineer at Intel Corporation based in Penang, Malaysia and is responsible for software release qualification and process improvement. Prior to this role, he was a Software Test Engineer and Test Project Lead at Motorola Solutions responsible for two-way radio software related functional testing, test planning, defect prediction, escaped defect analysis and test project management. He was a Supplier Software Development Engineer in the same organization and responsible for supplier engagement in software development and process improvement. In these roles, his efforts focus on software product quality and business process improvement. He has held the Digital Six Sigma Green Badge since 2019 and holds a Degree in Computer Science from University of Bolton in the UK.

Chin Pei has 16 years’ experience in software engineering. She worked as a validation engineer, software development engineer and automation engineer in her previous company. Her extensive experiences in different software roles have helped her to contribute to the software product development, quality, and processes in Intel. Currently, she is a passionate Quality Engineer who is responsible for ensuring that software meets Intel’s software quality release criteria (including areas like legal compliance and security compliance). She provides software process assessment, trainings, consultation, and guidance to the teams across different functional and regions. She has submitted automated and Lean projects which have greatly helped the validation projects to improve ROI, reduce resources and cost.

Copyright Felix Eu, Tan Chin Pei 2021
1 Introduction

In this introduction, we discuss what testing is and the important of selecting testing techniques that determine which validation approach is best suited to the organization. Subsequently we introduce the risk-based validation approach and the benefits of using it.

In Section 2, we discuss the software validation transformation, and how the organization can transform from the traditional software testing approach to the customized risk-based validation approach. We cover various transformation factors which include risk analysis, defect prediction model, project monitoring and control plan and building up subject matter experts (SME). Section 3 builds on Section 2 and provides examples of source of impact analysis.

Section 4 describes the risk management process and the various ways of leveraging the product risks, summarizing the test prioritization based on business risk and typical defect detection trend. In section 5, we elaborate on the mapping of the validation tasks to the original impact analysis summary provided by software developers.

Section 6 describes our experiments and results obtained when compared to full regression testing. This section also covers the gap analysis and maintenance process with an end target of improving the current risk-based approach. In section 7, we provided the implementation results of risk-based validation approach. Finally, in Section 8, we summarize and provide directions for future work and areas to research.

1.1 What is testing?

What is testing or validation? According to ISTQB Certified Tester Foundation Level Syllabus, 1.1. What is Testing? [5] “Software testing is a way to assess the quality of the software and to reduce the risk of software failure in operation.” Testing is the process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.

1.2 Testing Techniques Selection

Selecting an appropriate testing technique has been the key element of fulfilling the testing goal. It is obvious that appropriate testing techniques should be used in conjunction with the process of dealing with testing constraints and risk analysis. In general, both risk analysis and test constraints play important parts to ensure the right testing technique is selected. We need to prepare an appropriate methodology, putting in all the test constraints and risk factors, and perform simple evaluation to check if each of the available testing technique works and can achieve the testing goal. This is to avoid time wasted due to inappropriate testing technique being selected.

Let’s review the examples of test constraints in Figure 1. To achieve effective testing, we should test everything we can to obtain full test coverage so that we can find as many defects as possible. In contrast, to be efficient we should minimize the testing time and only perform certain tests, and we might not be able to find as many defects as possible. As you can see in both scenarios, the test constraint prevents us to achieve the efficient and effective testing.
Figure 2 represents a high-level fundamental view on risk analysis in conjunction with the testing goal and proposed testing technique. The idea is to further drill down the testing scope by evaluating the possible risks by asking a list of right questions like for example “If I only have limited time/resources to perform the validation”, “Why I should perform full validation if the code changes are on specific areas”, “Is there expectation for defects to be found if the same validation strategy is performed from one cycle to another cycle” and so on, this can narrow down the selection of testing technique.

Apart from test constraints, other risk factors can also be measured to determine the appropriate testing technique selection for the project. In fact, there may be a combination of different test selection techniques in a single project. The depth and coverage of each of the testing techniques may vary from one to another. For instance, during the initial project execution phase, we may want to take fewer risks and perform full regression testing to ensure all the newly developed code and functions have been thoroughly validated. Towards the end of the project when time constraints are more critical, we may want to change the test strategy to reduce the test coverage by taking more risks which only specific validation will be performed based on risk assessment. Some will perform sanity test or smoke test just to ensure the basic functionalities are working accordingly.

So, if both the test constraint and risk analysis are considered in the evaluation of testing technique selection, the chances to choose the right testing technique are greatly improved.

### 1.3 Risk-Based Validation

The testing technique we’ll be examining here is the risk-based validation approach. In a critical situation when a software release timeline has been promised to the customer, it is crucial to ensure the entire software code is thoroughly validated. However, it is time consuming and not feasible to conduct a full regression testing whenever there’s a small change in code to ensure functional integrity of the software. Even with major code changes, not all code is updated due to the new implementation. The question that always been asked is how these code changes can be validated and what kind of validation strategy can be used to achieve the maximum validation coverage of the code changes. Risk-based validation is one of the testing approaches adopted for a project which utilizes the impact analysis done on software code changes, prioritizing tasks, and taking informed risks to perform the validation. A software development
team must provide a summary of software code changes in the format of impact analysis data based on the source code delta from previous software release cycle. In fact, they know very well which codes they have changed. The impact can be categorized as low, medium, high or any numbering symbol and presented in a table format which describes the relation of the software code changes between the software feature and specific function/area in a project (see Table 1). The output from the impact assessment shall be discussed in the project meeting, clearly communicated and aligned with the software validation team to determine the risk-based validation strategy used for test optimization. In addition to the software code changes, incorporating defect prediction outcome into the impact analysis will greatly enhance the test optimization through risk-based validation approach. This risk-based approach can be used whenever there is a limitation or constraint on time, cost and resources of a project and whenever there is a need to optimize the test resources. Nevertheless, the risks taken should be balanced with the Return of Investment (ROI) gained in a project.

The risk-based validation approach mainly focuses on the considerations below:

- Specific area of the product where there is a high impact on business due to failure or high likelihood of failure in the production.
- Early defects detection and allowing a team of software developers to fix them as early as possible
- Quality control to improve on 'end to end customer experience'.

2 Software Validation Transformation

The world of software validation has rapidly changed, and the trend indicates that it is transforming from the traditional validation to a different kind of more cost-effective and adaptable validation approaches including risk-based validation. The maturity of the validation process in an organization is the key determination for such transformation and can be measured and represented through CMMI levels. Organizations with higher maturity level tend to have focus on their process improvement efforts on a prioritized and manageable number of practice areas. The performance and process improvement achievements increase capability to level up their software validation practices. Considering the availability of the testing constraints and the process improvements in place, many large organizations have taken the precedent for the transformation in software validation approach.

The process for this transformation may include but not be limited to the following:

- Risk Analysis:
  - Risk identification and assessment
  - Risk control and mitigation
  - Risk communication and action
- Identify a defect prediction model
- Develop or enhance a monitoring and control plan
- Build up Subject Matter Experts (SME)

2.1 Risk Analysis

According to the Oxford English Dictionary, risk is “the possibility of something bad happening at some time in the future; a situation that could be dangerous or have a bad result.” In software validation, risk analysis is an approach to identify and analyze risks, implement a control and mitigation plan, and measure how the risks may create threats to the functionality, stability, security, or performance of the application. The result of the risk analysis will be communicated to the relevant party and the recommended action plan will be taken to avoid any factors that can bring negative impact to the business or project.

Figure 3 shows the Risk analysis activity model. This model is taken from Karolak’s book “Software Engineering Risk Management”, 1996 [6] with additional comments (in blue oval).
In general, risk analysis process happened throughout different phases in software development lifecycle. At the beginning of the project execution, team is asked to perform risk identification. This covered a wide area of development and validation activities from where potential risks are likely to happen with reference to the past projects. In validation, test conditions will be the focus and high-level risk assessment will be performed against all possible test conditions. Risk assessment output will be documented, and recommended risk strategy will be included in the test plan. Of course, risk assessment and risk mitigation are continuous processes throughout validation phase and will be constantly monitored to identify any new risks. For example, validation scope from the defect analyses is unable to be fulfilled due to insufficient human effort to support the large impact areas. In this context, test plan may need to be updated to align with mitigation plan. A summary risk report should be generated in a weekly basis in accordance to each of the test metrics such as validation passing rate, validation task completion rate, defect detection rate and so on. All the open risks should be revisited to determine the current occurrence probability rate and to predict any new risks.

![Risk Analysis Activity Model](image)

### 2.2 Defect Prediction Model and Potential Impact

Identifying and developing a defect prediction model requires a significant effort to collect, analyze, categorize and pull the defect data from a database into a presentable chart that shows the correlation of defects uncovered between historical projects and current projects. It is one of the most powerful software metrics to describe the relationship among the complexity of the software implementation, software development time and the probability of the code errors invented by software developer.

The defect prediction outcome provides the level of confident and quality assurance of the software application under development. It acts like a benchmark and quality standard to gauge the software performance by ensuring the actual defect arrival is consistently approaching the predicted number during the entire software development life cycle. To enhance further, the validation team may consider incorporating the defect prediction outcome to check whether the defect arrival rate is on-par with the defect prediction. Depending on the actual defect arrival rate, if the actual defect arrival and prediction number are not on-par, the software quality team can chip in and take immediate action to analyze why the actual defect arrival and defect prediction are misaligned. There are a lot of questions that can be asked to help to identify the root cause. If it is below prediction, software validation team can further analyze and determine if the right amount of validation has been conducted the right functional areas have been validated or the right validation strategy has been adopted. As a result, they can perform gap analysis and adjust the validation strategy to improve the test coverage.
2.3 Project Monitoring and Control Plan

According to the Project Management Body of Knowledge (PMBOK), “the Monitoring and Control Process Group consists of those processes performed to observe project execution so that potential problems can be identified in a timely manner and corrective action can be taken, when necessary, to control the execution of the project.” [12]

Organizations should have a project monitoring and control plan in place and enhanced as regular as possible to maintain their competencies in the business world. In fact, software validation also requires its dedicated monitoring and control plan. Test monitoring and test control focuses on software validation activities, efforts evaluation, risk analysis and the action plan and tracks its progress against the test metrics. Scope validation formalizes the acceptance of the final project deliverables. It ensures that the deliverables have been completed according to the plan and meet all the requirements set forth throughout the project. This is very useful when the limited test resources are available to perform risk-based validation.

The risk-based validation approach pursues a stricter monitoring and control plan as compared to full regression validation because of the test constraints. Though there is always buffer allocation during project planning, it takes more risks to execute the validation plan. The risk-based validation approach works very closely with defect prediction plan to reduce the risks. Regular audits on defect detection rate during the risk-based execution phase greatly helps to identify whether the right areas are validated, or the right SMEs are assigned to perform the validation. It helps to ensure that risks taken are reasonable and aligned to the project scope, budget, and timeline. Alerts will be promptly triggered for any unexpected outcomes that occur.

2.4 Subject Matter Expert (SME)

It is challenging to build and retain the competency of the subject matter experts. In general subject matter expertise can be built based on working experiences, talents, research, and study etc. so that SMEs can be trained to have an in-depth understanding of a job and are able to thoroughly and accurately discuss duties and responsibilities of a job, knowledge requirements, skill requirements, ability requirements, and other specific software feature knowledge.

The complexity of a software application doesn’t always come from a single feature implementation but from features from different entities, subjects and aspects - for example security, functionalities, performance, graphics, etc. Subject Matter Experts are required for various purposes during several phases of the development and validation. From software validation perspective, SMEs are needed in specific content areas to provide judgments on tests already written and validated by other SMEs. Judgments are needed for validation and setting performance standards for these tests currently being field tested.

Of course, challenges arise during the process of transformation. However, the consistency and commitment to meet the higher quality standard of software validation pays off all the efforts invested for this change.

3 Source of Impact Analysis

Impact analysis helps to identify the potential consequences of any changes made on software. While software code changes are unavoidable and these changes may come from various sources, such as business needs adjustment, new requirements, new technologies, etc., it is very likely that the changes could result in failure or become out of control. Impact analysis helps to oversee the risks of the change and the resources which we should plan forward for implementation of the change.

Defect fixing is one of the common activities in the entire software development life cycle that causes the software code to change. Prior to the commencement of defect fixing, each of the defects should be triaged and all the discussions and resolutions will be properly recorded and updated in the defect
tracking. This process will serve as proof of evidence on the general agreement of correct categorization of priority, exposure and feature of the defects. Once the defects which targeted in a predefined software release milestone are fixed, software developers are requested to enter all the impact data into the impact analysis table (Table 1) one by one and these data will be consolidated by the software program manager. Filling the impact analysis data shall be as precise as possible to avoid ambiguity. Once the software release candidate is finalized for the milestone, it will be officially communicated to the software validation team during the software handover meeting. The same process is repeated for software code changes due to new features implementation and new requirements added into the scope.

4 Risk Management and an Approach to Validation

Risk management is the process of identifying, assessing, and controlling threats to an organization's capital and earnings. Before we start on a new project, we must first perform risk identification through the risk screening process based on impact and probability. These two elements shall always be evaluated on different project characteristics. In most cases a mission critical project tends to take lesser risk as compare to normal project. The reason is risks that are characterized as both high impact and high likelihood of occurrence often cause a project to fail if it is continued despite the risks. However, it can be balance up by the potential benefits gained that taking the risks is justified.

The risks in developing the validation strategy can be reduced by taking the risk-based validation approach with proper analysis and assessment on the delta in the software code instead of repeatedly performing full regression testing. Experienced software test analysts or technical leads utilize their best knowledge to understand and interpret the data in the impact analysis table which represents the software code changes. Each of the entries in the impact analysis table will be thoroughly analyzed to capture all possible dependencies and interactions with other features. For example, software code change in Dockers containers may impact any features that have a dependency on Docker images. If the data have similar characteristics, they will be combined and consolidated. The output will be a list of tests or validation tasks that are specific to the overall software code changes. However, each test or validation task has its priority which has a correlation to the original impact when the defect is triaged.

Product risk level can be managed via risk-based validation in the following ways:

- Start the risk-based validation as early as possible to identify the correctness of the assessed impact and to determine if more validation should be performed.
- Prioritize validation in the high impact areas.
- Have mitigation and contingency plans in place to complete validation after uncovering high impact defects. For example, workarounds to continue the validation until all impact areas are covered.
- Implement measurements of how well the risk-based validation approach at finding and removing defects in critical areas.
- Conduct proactive continuous risk assessment on non-validation areas to ensure they are defect-free throughout the release cycles.

The goal of risk-based testing is not to achieve a risk-free project but to carry out the testing with best practices in risk management to achieve a project outcome that balances risks with quality, features, budget and schedule.

Figure 4 shows the risk-based validation aligns the validation activities with business priority and achieves optimal risk coverage with focused validation. The chart represents the situation where it rapidly reaches over 80%+ business risk coverage, using only 20% of your test effort [3]. Risk Coverage Optimization shifts the focus from test coverage to risk coverage.
Figures 5 denotes a healthy defect detection trend against test hours. Though there are a number of factors that could influence the defect arrival trend, a typical defect discovery trend normally has a sharp increase during the beginning of the validation process and then decreases after a certain period of time. We must grab the opportunity and keep the right validation to find as many defects as we can during the golden hours. When the curve flattens, it is not worth to continue the validation process as the defect density is almost at its maximum limit. Though regression testing may be able to uncover most of the defects eventually however it may be too late with no specific focus in the validation process. Risk-based validation functions in such situations aim to flush out as fast and as many defects as possible while maximizing the test coverage.

The risk-based validation approach will prioritize tests based on the impact assessment from the overall software code changes. Depending on various scenarios and situations, we can then optimize the right amount of validation to be performed on one feature over other features and we can include feature interaction tests. Both software development and validation teams play critical roles to ensure the accuracy of the software impact given (input) and the tasks defined from the impact analysis (output).

5 Validation Tasks Traceability

The efficiency and effectiveness to interpret and convert impact analysis data into a list of validation tasks is the key success of the risk-based validation approach. Defect tracking systems such as Jira, Bugzilla, HP ALM, Mantis and so on are great tools for defect management and new features implementation. Impact analysis activity requires a powerful defect management tool to improve its accuracy and completeness. The impact analyses we have discussed so far focus on the outcomes of the software
code changes where each of the outcomes are labeled with a respective “impact categorization” and precisely plotted into a two-dimensional table as shown in Table 1. Each of the entries denotes correlation between features that have code changes. In Table 1 below, the determination of the high impact was observed on Feature-1 alone so it is marked as 1 (High) in the cell that correlation between Feature-1 and Feature-1. Next, code changes happened between Feature-1 and Feature-2 however the impact is average so it is categorized as 2 (Medium). In fact, the impact analysis table aims to enhance process integrity and provide clarity to all the impacted features.

![Table 1: Impact Analysis](image)

From the impact analysis table, the intention of the software code changes has been gracefully transformed into systematic, well-defined, and testable tasks in Table 2. For example, the intersection of Feature-1 and Feature-1 has an impact of 1 (High); this mean only Feature-1 alone is impacted and requires higher validation coverage. As another example, at the intersection of Feature-1 and Feature-4 there is an impact of 3. This mean both Feature-1 and Feature-4 are impacted but the impact is 3 (Low), so only minimum validation coverage will be planned for Feature-1, Feature-4 and the interaction between Feature-1 and Feature-4. Further analysis shall also be performed to check if the impact is only happened on specific scenario, situation, or environment.

Each test or validation task must provide a traceability capability so that unrelated tests can be eliminated. Effort can also be calculated from the analyzed impact which allows software validation team to optimize test coverage with optimal effort based on the availability of test resources and prioritization of the validation tasks.

**Table 2: Validation Strategy for Milestone X**

![Table 2](image)

Although it looks easy to conduct the impact analysis by turning them into more specific tests, obviously the efforts behind to perform such analysis is tremendous. In the previous sections of this paper, we explained that risk analysis, developing a control plan and building up subject matter experts all are key contributors to the success of such impactful activity.

### 6 Gap Analysis and Maintenance

Over the years, much research has been carried out to identify the root causes of software failures. One of the main reasons for such failures turned out to be poor quality assurance during the software development process. The main purpose of executing stringent quality assurance tests is to prevent the release of poor-quality software products. Small mistakes that slip through may potentially lead to large
financial losses. That’s where a gap analysis comes in. It provides an objective perspective to improve processes for the future.

Gap analysis takes into consideration of different aspects surrounded quality. It is important to allocate appropriate time and depth to a review of the gap analysis. The gap analysis may identify situations where risk-based validation has never been performed or may find that risk-based validation performed is lacking in some critical details. The outcome allows us to prioritize testing efforts and detect defects early in the release cycle instead of in production. However, it is important that the effectiveness of any gap analysis lies with the technical competence of those performing the analysis. Unless those performing the analysis are technically competent and have good attention to detail traits, otherwise the effectiveness of risk-based validation that have previously been performed cannot be truly determined.

Quality reviews shall be performed on the risk-based validation approach on a regular basis to identify potential quality gaps as below:

- Impact analysis data is not available at the right time, or it is delayed
- Incorrect entries in the impact analysis table
- Misalignment between defect prediction and defect arrival trends causing misunderstandings to different parties
- Inaccurate risk assessment causing the wrong target for testing
- Ambiguous or wrong definition of the validation tasks which is unable to be matched to the impact analysis data
- Improper prioritization of the validation tasks
- Duplicate entry of validation tasks which can be combined and consolidated
- Insufficient test coverage on extremely high impact area
- Redundant test coverage on extremely low impact area

A small change in software code may have a big business impact to the organization because of the additional resources required to support it and the changes required to go through the entire software development lifecycle again. A gap analysis has a direct impact on an organization’s efficiency, which in turn affects the bottom line. With the insights gleaned from the gap analysis, the respective improvements can be quickly implemented to boost performance for future activities in the following ways:

- Turn the finding into action plans which will fill in the gaps
- Develop plans to address the gaps
- Assign and track the action plans until closure of the gaps.

There are no perfect testing methodologies that can ultimately produce a defect free software product. Without exception, risk-based validation approach will still require having regular quality reviews in place to improve all the identified gaps. The aim is to maintain risk-based validation as one of the most competitive approaches to meet an organization’s needs, utilizing all the possible best practices to leverage the risks, quality and all other resources. The best way to create high-quality software product is to implement effective quality management that provides tools and methodologies for building defect free software products.

7 Implementation Results

We carried out measurements between the traditional full regression testing and new risk-based validation approach on different aspects such as staff effort, test coverage, test cycle time and test efficiency. We found that with the risk-based validation approach the test efficiency has been improved by at least 50% with the same test coverage but spent only approximately 50% of the staff effort and reduced test cycle time by approximately 30%. This is because the validation scope is narrowed down, avoids repetition, and concentrates on code change areas rather than validate the whole test suite. As a result, a shorter test cycle time is required to complete the validation for the targeted milestone.
coverage is accumulated over different milestones and will eventually achieve 100% of the planned validation. However, the risk has been increased by 20% but this is the nature of the risk-based validation approach and it is paid-off by the ROI gained.

8 Conclusion

In this paper, we elaborated the inefficiency of traditional regression testing where time, human efforts and early defect detection are concerned, and we improve it to a more efficient way of doing testing. We aim to enhance the risk-based validation approach which is essential to optimize the test coverage with specific focus on risks analysis, task prioritization, defect prediction trend and decision making.

Risks should be properly tracked throughout the software product life cycle (SPLC). They should not be limited to software code changes but should also include all aspects that may create threats to the software validation and product release. It is no surprise that validation activity will be badly impacted when management in an organization cuts both budget and time for a software project. It is critical to use the correct skillset to turn it into success. The general methodology for this situation is not to test everything a little, but to concentrate on high-risk areas and the most defect-prone areas.

The analysis of the impact-based risk assessment is critical to the success of the project commitment and focus on minimizing the escape defects to the fields. It helps to optimize and attain complete productivity which in turn increases the results, efficacy, and efficiency of the system.

Imagine how much benefit we can gained with risk-based validation approach! The results show that it required a shorter test cycle time to complete the validation for the targeted milestone while maintaining the same test coverage and the increased risk of 20% will be contra with ROI gained. The ultimate goal in performing the risk-based validation is to achieve a project outcome that balances risks with quality, features, budget and schedule for the time, resources and efforts that have been invested in a program.

References

10. CMMI Levels of Capability and Performance, https://cmmiinstitute.com/learning/appraisals/levels