Risk-Based Testing

- Overview
- Risk-Based Testing (RBT)
  - Purpose
  - RBT at different levels
  - RBT Process
    - RBT in Waterfall and variants
    - RBT in Agile
  - "Traditional", RBT and Hybrid RBT approaches
- References

Overview

This article provides an overview of Risk-Based Testing (RBT).

RBT is built on top of Risk Management and thus the same concepts and principles apply.

Learn more

If you want to learn how to perform RBT whenever using Xray, please have a look at Performing Risk-Based Testing (RBT) with Xray

Risk-Based Testing (RBT)

RBT (Risk-Based Testing) is a testing strategy that follows the principles of Risk Management but in a testing context.

In software development projects, or in any development project in general, there are a set of common objectives among these are:

- "quality"
- cost control
- time-to-market (i.e. target release/delivery dates)

These goals can be affected by risks coming from different sources (internal or external,) due to all sorts of factors. In case of negative risks, the risk level will depend on things such as the complexity of the change, number and kind of impacted users, frequency of usage, likelihood of failure, etc.

Risks can affect the overall quality, including the customer perceived quality.

Quality is a broad word embracing "fit" of the product to customer needs and many quality attributes (i.e. "quality properties" as per ISO/IEC 25010:2011), which risks may affect. An intrinsic part of quality is ensuring that a product has few bugs (or at least a reduced probability of bugs on important/risky areas).

If testing is performed as a means to find/avoid bugs, RBT can be used as a "mitigation", or even as an "avoidance" strategy.

On the other hand, testing is also about understanding, discovering and providing valuable feedback that can help build "better" features and better products. In that sense, RBT may be used to perform thorough testing or doing it lighter/roughly depending on the risk.

Thus, RBT can be seen as a strategy that uses risks as means for adequate testing (depth and priority) in order to maximize testing goals. It will affect all testing activities (e.g test planning, design, execution).

Therefore, RBT assists in making testing related decisions based on the assessment of risks.

Purpose

The main purpose of RBT is to use risk management principles for adequate testing.

First of all, it provides a framework for clear communication and discussion of risks, by defining terms and a common language; it also makes risks visible and actionable.

RBT covers customer needs and also the needs of the development team, using risks as the input to support testing activities.

Customers are mostly concerned about business features, timing, visible quality and costs.

The development team has similar concerns; however, it sees quality in a broader scope as it has to maintain and possibly evolve with the product being developed. Timing and costs need to be managed effectively to be on a budget and avoid delays; however quality must not be hurt and if decisions need to be taken to meet timing/costs criteria, RBT can provide the means to ensure that focus will be given to the features/issues that matter most to customers.

Both customers and the development team want to avoid important defects, thus RBT focuses the testing efforts on what matters most - where we can get most value from.

Overall, RBT's purpose and benefits can be summed up as:

- **Focus more on the customer**
  - test more thoroughly what customers are most concerned with
    - deliver what is most important for the business
- **Reduce the probability and impact of negative risks**
  - by focusing testing on higher, negative risks, probability of missing important defects lowers and therefore the probability assigned to the risk lowers as its corresponding risk level; on the other hand, as testing also provides feedback to the team including its developers, the impact of a certain risk may also decrease as the feature being worked out may be done differently or better
- **Increase confidence**
  - RBT can help find more important defects first, by focusing testing on higher risks (paper)
  - by focused testing on higher risks (or risky areas/features), RBT ensures that important items are exhaustively tested and important defects are found sooner; thus, product and it's most important items (e.g. features) can be released with a high degree of confidence, while ensuring they meet expected quality goals
- **Optimize QA effort and cost**
  - RBT can answer questions such as...
    - What should we test? Everything? But we don't have time...
    - Where should I start?
    - When can we stop testing? We have to make the release and move on...
    - Can we reduce the testing effort somehow? How and at what "cost"?
    - If we have more time for testing, what is the best way to take advantage of it?
  - RBT provides the means to define the testing scope, focusing on what is relevant, by identifying what tests to execute and their execution priority, given time and resource constraints; the overall amount of test cases may be reduced as not everything needs to be tested or be tested in the same depth
  - RBT provides a way (based on risks) to choose tests for regression testing
  - RBT provides some clues for selecting candidates for automation
- **Increase risk level of positive risks**
  - If an opportunity is identified, RBT can be used to provide thorough testing and take advantage of it. If a feature is being implemented and if the team finds that making it slighter differently, perhaps by generalizing it further or making it more visible, it can increase the probability of positive risk, as end-users may start using it for additional scenarios and thus increase the product added value. Using testing as a constructive feedback loop, knowing the opportunities and their relative relevance, can increase the likelihood, impact and the overall risk level for opportunities

- **Make a release go/no go decision based on risks**
  - sometimes you may need to ship the product sooner, for time or budget constraints
    - RBT can give you visibility of risks, so you can take a go/no go decision "knowing the risk"
  - RBT can be used to overcome risks that block "acceptance" (i.e. customer acceptance of the release)
  - RBT implicitly explains why certain tests were executed over other ones, and thus ease communication with other stakeholders and avoid discussion on why some tests were not executed at all

RBT at different levels

Risks may be evaluated at different levels:

- **at project level**
  - missing cross-functional knowledge on team
  - outsourcing IP
- **at "requirement" (e.g. Story) level**
  - open-source library support
  - performance degradation in concurrent scenarios
- **directly at Test level**
  - validation of specific scenarios/use cases whose relevance is different from other ones, in some context (e.g. on a specific release)

Many teams perform RBT identifying and evaluating risks at "requirement"/Story level: thoroughly testing is performed on risky "requirements", while less risky "requirements" are tested in a more pragmatic way.

However, sometimes "requirements" may not exist at all, defined explicitly as such. This may be common in scenarios where teams inherit legacy projects, that have few or no "requirements" but that have Tests that validate the intended behaviour.

	Project level	"Requirement" / User Story level	Test level
Pros	• some risks may simply be project wide	• most common approach • allows deciding whether a story should be addressed and how based on its risk • shapes the test's definition on the Test Design phase taking into account the risk	• allows distinct prioritization of Test cases for the same requirement • allows RBT in projects without requirements (e.g. legacy projects)

Cons	• misses a more granular way of assessing risks and addressing them	• not differentiates between Tests for the same story, although some additional criteria (e.g. ”priority”) could be used for that • not applicable to projects without requirements	• a lot of effort to evaluate Risk for tons of test cases • Tests are not specified taking into account requirement risk; applies only to existing test cases and not to new ones (at the acceptance criteria would probably be a better approach)

RBT Process

RBT is much more than selecting what Tests to execute and their order; RBT affects test planning, authoring and execution.

You may implement RBT in several different ways, formalizing the process a bit more or not.

The overall process involves the following macro-activities.

1. **What and how "things" can affect our project?**
    a. **Risk Identification**
        i. identification of risks to product (e.g. software/hardware) features; this is normally performed by the team and discussed together
        ii. risks are identified before implementation starts and are reviewed throughout the development life cycle; new risks can be identified during implementation
    b. **Risk Analysis**
        i. risks are discussed together in the team (may involve customer)
        ii. risks impact and probability are calculated, and thus the related risk level
2. **How will we handle it in the most effective way (i.e. what can return the most value)?**
    a. **Risk Evaluation and Treatment**
        i. **Test Planning**
            1. using the input from Risk Analysis, the test manager can...
                a. define test strategy (e.g. level of testing to perform, techniques to use, environments to choose)
                b. estimate testing effort
                c. define/estimate schedule (target dates, number of testing iterations)
        ii. **Test Design**
            1. specification of the Tests taking the identified risks as input; tests are designed to mitigate the risk (i.e. diminish their probability)
            2. make use of more extensive data with data-driven testing and automated testing/checking, if needed
        iii. **Test Execution**
            1. perform testing by descending order of risk level (i.e. execute Tests related to higher risks first); experienced testers, with a high degree of domain knowledge, may provide more valuable feedback
            2. thoroughly test risky items, using scripted and exploratory approaches
    b. **Risk Monitoring and Reviewing**
        i. look at items where you assessed the risk and evaluate if you need to take additional measures/treatments ("Are those items having failed tests? Were bugs found? Are those bugs relevant? Did we find any new risk while testing?")

RBT in Waterfall and variants

In projects following waterfall, or one of the waterfall variants, in their SDLC, RBT can fit as followed: (high-level, simplified overview)

1. Risk Identification & Risk Assessment
    a. BAs together with the team discuss risks in the ”Requirement Analysis” phase
2. Risk Treatment
    a. performed by testers, during the “Testing” phase
        i. Test Planning takes the input of Risk Assessment to define the testing strategy and effort; the testing scope is agreed
        ii. if adopting the scripted testing approach, testers start by exhaustively detailing test cases for the riskiest items, during Test Design
        iii. testers will perform more in-depth testing on the risky items during Test Execution
        iv. if important bugs are found on the risky items, the software can go back to "Implementation" phase as-soon-as-possible
3. Risk Monitoring and Reviewing
    a. throughout the project development life cycle; special focus is given at testing closure,

Please note that even though you may be doing Risk Identification and Assessment on the first phase (i.e. "Requirement Analysis,") risk factors are present in all development phases as each one is prone to risks.

As your development project goes through different stages sequentially, problems found at a later stage of the pipeline can lead to rolling back to a previous phase; thus, it can go from Test Execution back to Test Planning... or from the "Testing" high-level phase back to "Implementation"... or even from "Implementation" back to "Design" and "Requirement Analysis.

RBT in Agile

In projects adopting Agile principles and values software is delivered incrementally and frequently, and highly focused on customer needs and feedback. In Agile, changes are welcome and not really tied to "long phases." Both Agile Software Development and RBT have the customer and their most important needs/concerns at the center.

A possible implementation of RBT in Scrum based projects could be as follows.

1. Risk Identification and Assessment
    a. Release planning
        i. Highest-risk tasks must be placed in the earliest sprints
    b. Backlog refinement/grooming
        i. the team can identify and discuss risks together on each PBI (product backlog item)
    c. Sprint planning (by the whole agile team)
        i. Testers can start creating a risk-based analysis by using the stories given for that iteration, identifying the high-risk stories that can be tested on that iteration, and prioritizing the user stories depending on their criticalness to the business
        ii. Risk's impact and likelihood can be evaluated using "Risk Poker" (like planning poker)
2. Risk Treatment
    a. During sprint lifetime, by performing more extensive testing on high-risk stories (including for regression testing purposes)
3. Risk Monitoring and Reviewing
    a. During sprint lifetime

"Traditional", RBT and Hybrid RBT approaches

While in "traditional" testing, tests are executed in "ad-hoc" way (without any special criteria/order), in RBT risk is used to choose what tests are to be executed and their order of execution.

RBT also conditions how testing is done (and how test cases are specified): focus is given to tests associated, directly or indirectly, with and higher risk.

In Hybrid RBT, non-risk-based Tests (i.e. tests specified without considering risks) are executed by the risk descending order.

References

- Risk management - Principles and guidelines: ISO 31000:2009(E)
- Souza, Ellen Polliana & Gusmao, Cristine & Venâncio, Júlio. (2010). Risk-Based Testing: A Case Study. 1032 - 1037. 10.1109/ITNG.2010.203.
- RISK FACTORS IN SOFTWARE DEVELOPMENT PHASES, Haneen Hijazi, Msc Hashemite University, Jordan Shihadeh Alqrainy, PhD; Hasan Muaidi, PhD; Thair Khdour, PhD; Albalqa Applied University, Jordan
- What Techniques Can Be Used for GUI Risk-based Testing?, Behzad Nazarbakhsh and Dietmar Pfahl ,Institute of Computer Science, University of Tartu
- James Bach on Risk-Based Testing: How to conduct heuristic risk analysis, James Bach, 1999
- ISO/IEC 25010:2011 and old ISO 9126:1999 (brief, non-official overview)