Name: Thien Tran
ID#: 1222245570

CSE 565: Software Verification and Validation
Project 3- Part 1 Report
Oct-09-2021

Assumptions

This project assumes that the contingency plan includes the risk mitigation plan as required during the live event; not vice versa per standard practices based on the Project Management Institute guideline.

The quality of documentation affects the speed of tasks related to testing and fixing.

The availability and the reliability are grouped into one group: reliability and availability. Not much information is explicitly mentioned nor can we statistically quantify clearly one nor the other.

Since the case states that a quarter of the code was built upon the legacy code and there are 3 portions following up. The project assumes that the size of the code is equally distributed into 4 portions).

Testing intensity is risk-based. This means that testing intensity is equivalent to the risk exposure of each component.

Risk prioritization

This project has condensed the given information into a table [table 1]. Since the severity of a failure of each component is not explicitly said, this project will proceed to establish a set of business rules based on Srikantha's & Banerjee's severity levels (p. 1181) as the basis of estimate (appendix 2). This project establishes that the legacy code and the new code sections belong to the catastrophic severity level, the API section belongs to the critical severity level, and the Podcast section belongs to the marginal severity level.

Once the severity level for each component is established, this project proceeds to output the consequences of failure of each component, the consequence is a function of

likelihood and severity given in the project 2 part 1 description, risk exposure = f(likelihood, severity) (appendix 3). The result is as followed (appendix 4):

	Risk exposure
Legacy code	High
Podcast section	Low
New code	Medium
APIs	Medium

The risk exposure is then used to determine the priority of testing.

The risk exposure can be used to determine the ranking. However, since the new code and API have the same risk exposure level.

Risk prioritization

	Ranking
Legacy code	1
New code	2
APIs	2
Podcast section	3

Test intensity

	Intensity
Legacy code	Maximal
New code	Medium
APIs	Medium
Podcast section	Low

Contingency plan

Contingency plan includes 2 basic strategies: risk prevention and risk minimization.

Risk prevention applies to risks which have not manifested. A risk doesn't occur suddenly. The risk becomes a consequence after we fail to recognize the symptoms of the cause, to determine the cause, and to eliminate it. This means that we have to have a process to prevent the risk from a cause from manifesting into a consequence. Therefore, we prevent a cause by inspection and risk monitoring and controlling. While the risk monitoring and controlling process seeks to eliminate the cause before too late, the inspection method and the automation of critical tests prevent the cause from lurking into the code production.

Risk prevention includes: risk monitoring & control, inspection, automation of critical tests, required combinatorial coverage.

Risk minimization strategies are applied when a risk has already manifested. Its goal is to minimize the loss of net revenue of consequences occurring from the risk manifestation.

Risk minimization includes: risk sharing, risk reduction, risk acceptance.

Should one or more risks occur at the same time, the order of fixing is based on the risk prioritization table.

The strategies are applied to each component as follows:

	Required combinatory coverage	Risk Prioritization	Risk monitoring control	Inspection	Roll back	Automation of critical tests	N version Programming
Legacy code	High	1	Yes	Yes	Yes	Yes	Yes
New code	Medium	2	Yes	Yes	Yes	Yes	No
APIs	Medium	2	Yes	Yes	Yes	Yes	No
Podcast section	Low	3	Yes	Yes	No	No	No

Risk prevention strategies

Risk monitoring & controlling

Statistical process control

Manager establishes the statistical distribution of the defect removal efficiency (DRE) metric.

DRE = Defects found by the testing team / (defects found by the testing team + defects reported by customers)

Manager determines whether the DRE is within the limit bounds. If the DRE score deviates, manager proceeds to perform the root cause analysis to determine the desirable causes.

Root cause analysis

Via poisson distribution analysis, manager gathers defects reported by users to analyze the probable causes by asking 5 whys.

Manager identifies the most common causes which are considered root causes.

Manager seeks solutions to eliminate the causes.

Manager then proceeds to perform process improvement to bring back the DRE score within the statistical bounds.

Process improvement

Manager characterizes the problematic processes from the root cause analysis .

Manager analyzes the processes.

Manager characterizes the desirable states to upgrade the processes based on the result of the root cause analysis.

Manager redesigns processes.

Manager implement the new design.

Inspection

Daily inspection of each code section uploaded by the developers.

The inspector role is assigned daily to different developers. Inspector receives an inspection agenda from the superior manager as the goal. Inspector then gathers the items to be inspected from the supporting documents. Inspector carries a list of submitted components to be inspected and a checklist. Inspector signs off the code ready to be released once having finished the inspection.

Automation of critical tests

When a code section is uploaded, it is automatically tested by a set of tests that follow the critical requirements of the software.

Required combinatorial coverage

High level: multiple condition/decision coverage.

Medium level: condition/decision coverage.

Low level: decision coverage.

Risk minimization strategies

Risk sharing

N version programming

Developers develop different programs of the critical components, if one version fails, the other will take control over the processing.

Risk reduction

Version rollback

If the program crashes, the program will revert back to the last working version. A report will be generated and sent to the database.

Risk prioritization

To increase the software reliability and availability, the risks that impact the usability of the users will be prioritized based on the priority given prior.

Risk acceptance

There is nothing we can do to minimize the risk. However, we will put it into the feedback loop in the risk management register and lessons learned.

Appendix 1: Summary table

	Documentation	Compatibility issue	Time to test	Time to fix	Experienced	Likelihood	Severity (Appendix 2)	Quality of code
Legacy code	Poor	Poor	Slow	Slow	N/A	High	Catastrophic	N/A
Podcast section	Well	N/A	Fast	Fast	Experienced	Very low	Marginal	Good
New code	Well	N/A	Fast	N/A	Experienced	Very low	Catastrophic	N/A
APIs	N/A	Compatible	N/A	N/A	N/A	Very high	Critical	N/A

Appendix 2: Severity levels and the categorization of the severity level of failure of each component

Severity levels (p. 1181, Srikantha & Banerjee)**:**

Catastrophic: when a failure impacts the ability of a customer to use a product or the testing plan can no longer continue.

Critical: when a failure exists but there is a workaround and the product can continue to function with the workaround.

Marginal: when a failure exists and can be fixed in a later version.

Analysis of severity

Based on the information given and the rules of severity, the project categorize the severity of failure of each component into the following:

	Description (Project 3 part 1 description)	Severity
Legacy code	If there is an error, it will bring down the application for multiple days, rendering the site useless	Catastrophic
Podcast section	There would be very minimal damage if the section was not working for some time, and it would not affect usability	Marginal
New code	Other portions of the application could not be used, which would heavily affect usability and availability of the application	Catastrophic
APIs	If one of the APIs goes down, the application would instead take the local data it has stored for the API to keep the application running. The only downside would be that performance would be slower and users would not get the latest information. The consequences are not as severe as if the legacy code was down, but are more severe than if the podcast section was down.	Critical

Appendix 3: Risk exposure formula (Project 2 part 1 description):

Probability \ Consequences	Marginal	Critical	Catastrophic
Low	Low	Medium	Medium
Medium	Low	Medium	High
High	Low	Medium	High

Appendix 4: Risk exposures of the failures of each component

Given the likelihood (appendix 1), the severity level (appendix 2) and the function of risk exposure (appendix 3), the result of the risk exposures of each component is

	Likelihood	Severity	Risk exposure
Legacy code	High	Catastrophic	High
Podcast section	Low	Marginal	Low
New code	Low	Catastrophic	Medium
APIs	High	Critical	Medium