hermagst jonahhal simonamo dafolvel

Part 1

Test Scenario ID	1	1	1	2	3
Test Scenario Description	Use customer care page	Use customer care page	Use customer care page	Use admin page	Transfer funds
Test Case ID	1	2	3	4	5
Test Case description	Fill out and send customer care form	Fill out and send empty customer care form	Fill out and send customer care form with invalid email	Admin updates application settings from admin page	Transfer funds from one account to another
Test steps	1. Go to https://parabank.parasoft.com/parabank/index.htm 2. Navigate to the “Customer care” page by clicking on the ‘mail’ icon. 3. Fill out form valid form 4. Click "send to customer care" button 5. Verify by getting success message	1. Go to https://parabank.parasoft.com/parabank/index.htm 2. Navigate to the “Customer care” page by clicking on the ‘mail’ icon. 3. Do not fill out form 4. Click "send to customer care" button 5. Verify by getting error message	1. Go to https://parabank.parasoft.com/parabank/index.htm 2. Navigate to the “Customer care” page by clicking on the ‘mail’ icon. 3. Fill out form with invalid email 4. Click "send to customer care" button 5. Verify by getting error message	1. Go to https://parabank.parasoft.com/parabank/index.htm 2. Navigate to "Admin page" by clicking on the text to the top left. 3. Nagivate to "Application settings" section 4. Update new valid values 5. Click "submit" 6. Verify by getting "Settings saved successfully." message	1. Go to https://parabank.parasoft.com/parabank/index.htm 2. Login to account 3. Navigate to "transfer funds" 4. Enter desired amount 5. Choose sending and recieving account 6. Click "transfer" 7. Verify successful transfer
Precondition	• Parabank website is online	• User have registered user credentials • User is logged in with their account • Parabank website is online	• User have registered user credentials • User is logged in with their account • Parabank website is online	• Parabank website is online • User is a admin with registered admin credentials • User is logged in with their admin account	• Parabank website is online • User have registered user credentials • User have logged in • Have more than one account under one registered user
Post conditions	• Form has been sent to customer care. • Success message has been emitted	• Error message has been emitted	• Error message has been emitted	• Application settings have been updated • Success message has been emitted	• Success message has been emitted • Funds have been transfered
Expected result	Receive message "Thank you "name" A Customer Care	Receive error message	Receive error message	Recieve success message	Recieve success message

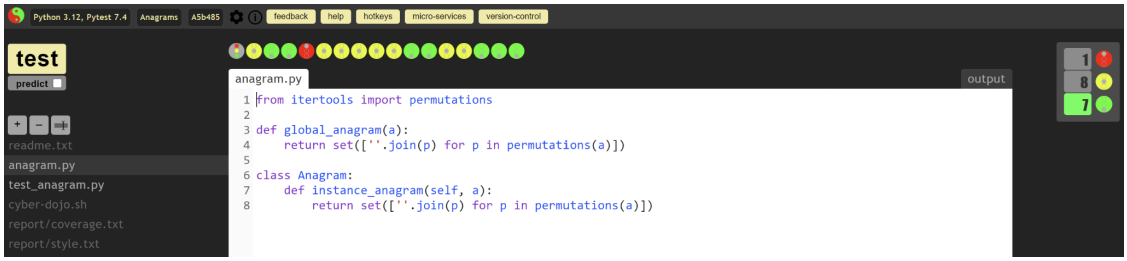
	Representative will be contacting you."				
Actual result	Receive message "Thank you "name" A Customer Care Representative will be contacting you."	Receive error message	Receive message "Thank you "name" A Customer Care Representative will be contacting you."	Recieve message "Settings saved successfully."	Recieved message "Transfer Complete!"
Status	Passed	Passed	Failed	Passed	Passed
Comments			Form should not accept invalid emails		

2.1 Benefits of independent automated tests

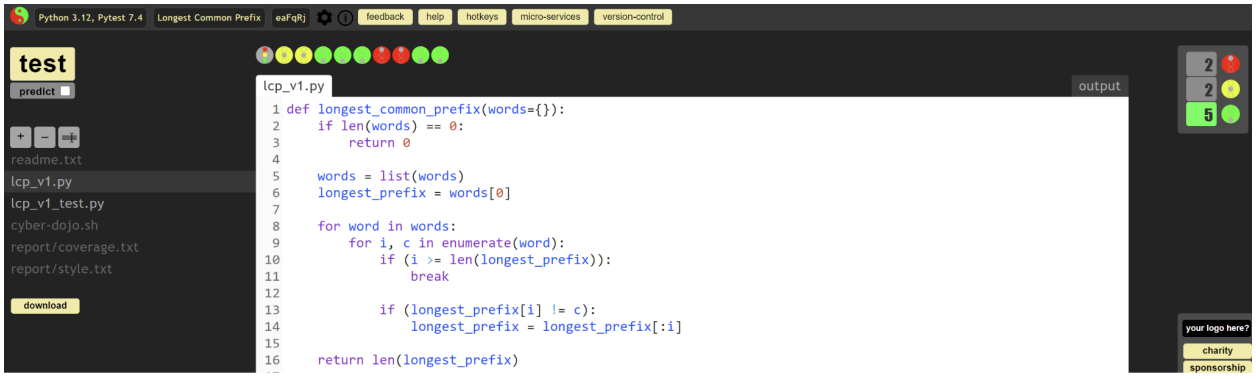
Consistent Starting Conditions:
Having each test start with the same state, reduces the chance of a test failing due to unexpected states left by previous tests.
Load Testing:
By running multiple tests simultaneously, you not only test functionality under load, but can also identify performance bottlenecks and concurrency issues which would not be apparent when running tests sequentially. This gives insight to how servers might behave in a more realistic setting.
Simpler Test Code:
Since each test is self-contained, there's less need for complex setup or teardown scripts that prepare or clean up after tests that depend on each other. This makes individual tests easier to write and understand.
Ease of maintenance:
Changes in the applications functionality are less likely to require widespread changes in the test suite. Procedures like login and logout are written outside of the independent test-suite.

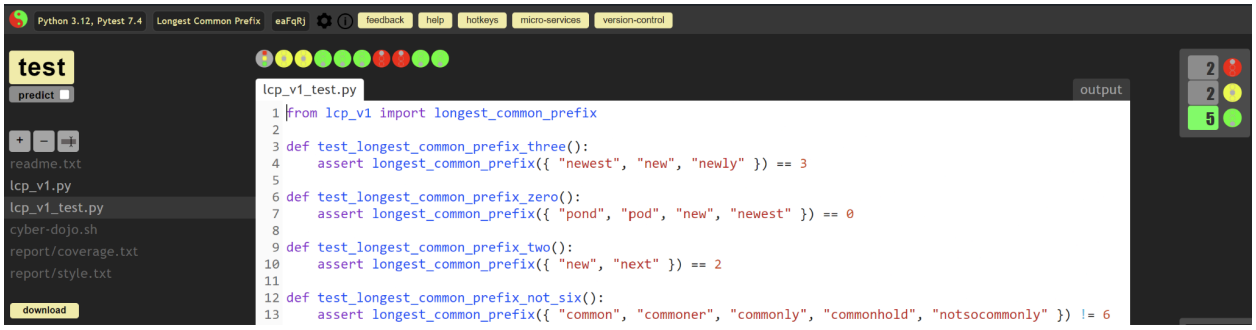
Part 2

Programming task 1 - Anagram (easy)



Programming task 2 - Longest Common Prefix (difficult)





Part 3

Task 0 & 1

We chose option 1 going with the amorphie workflow project. The Quality gate status for this project is *failed*, meaning the static analyzer has flagged the code base as failing in too many areas.

1.2

1.2.1 Bug: A coding error that will break your code. This needs to be fixed immediately.
1.2.2 Vulnerability: A point in the code that's open to attack by a user.
1.2.3 Code Smell: A maintainability issue. It makes the code confusing and difficult to maintain.
1.2.4 Blocker: Most severe error type.
1.2.5 Coverage: The amount of code that has been tested.

1.3

Data gathered at 2024-04-01 15:27
1.3 directory: /amorphie.workflow
1.3.1 21 bugs
1.3.2 0 vulnerabilities
1.3.3 203 code smells

Task 2

2.1 & 2.2

- Maintainability issue: `Change the visibility of 'GateWay' or make it 'const' or 'readonly'`. If a variable is not modified it is good practice to declare it as constant so that if someone tries to modify this variable in the future they know that this may have unforeseen consequences elsewhere in the code. To fix this add the `const` or `readonly` qualifiers to the variable.
- Maintainability issue: `Make this field 'private' and encapsulate it in a 'public' property.`. Unless a field in a class is used from outside the class it is good practice to declare it as private to keep the abstraction layer of that class intact. To fix this add the `private` qualifier to the variable.
- Adaptability issue: `Refactor this method to reduce its Cognitive Complexity from 17 to the 15 allowed.`. Functions that have too many conditionals and indentation can be hard to read as it’s hard to follow the flow of the function. To fix this make abstractions or draw parts of the logic out to separate functions to fix the issue.

- Maintainability issue: Remove this unread private field 'daprClient' or refactor the code to use its value. When a private field is written but never read it turns into a case of “dead store”. To fix this issue you have to either remove the variable if it is not used anywhere, alternatively if it’s a throwaway variable the convention is to rename it `_` (underscore) to indicate it’s not used and that that is fine. Finally, you can find somewhere to use the variable.

Task 3

3.1 & 3.2

False positive - when the desired outcome is negative but the actual outcome is positive. An example of this can be flagging a non spam email as spam when trying to filter out spam email.

False negative - when the desired outcome is positive but the actual outcome is negative. An example of this can be flagging spam email as real email when trying to filter out spam email.

Part 4 - Reviews as a test technique

Introduction

Reviews are an essential part of software development. They are necessary both to ensure the code is of a certain quality, to improve the communication between the development team and for the developers to learn and improve from both their own and others' mistakes. In this part of the assignment we will discuss the how, who, what and why of reviewing in software development. We will explain the different types of reviews and when to apply each of them, how to optimize them, discuss their usefulness, and much more.

A Formal Review process

The process begins with the author (the person responsible for the creation of the work product) submitting it to the review leader or facilitator. This submission includes all relevant documents or code that need to be reviewed. The review leader, also known as the facilitator or moderator, conducts an entry check to ensure the submitted work meets basic criteria for review. This check might assess completeness, adherence to basic standards, and suitability for review.

If the entry check is passed then the author and review leader work together to finalize which specific parts of the work product will be reviewed. This step ensures that the review focuses on the most critical or relevant aspects. At the same time, the review leader designates roles for the review process. Typical roles include the facilitator, reviewers who will inspect the work, the scribe, and possibly others depending on the methodology being used. The facilitator sets clear exit criteria that define what conditions must be met for the review process to conclude successfully. Exit criteria could include a specific number of defects to be identified, coverage of certain aspects of the work product, or the resolution of all major issues.

When executing a review, there are multiple fundamental steps that should be taken to ensure its success. Some of the fundamental steps for the execution of a formal review process during an inspection are *planning*, *initialization*, *issue & communication analysis* and *fixing and reporting*.

Initiate Review

The facilitator or review leader schedules an initial meeting, either in person or digitally. This meeting is crucial for discussing several key points. It provides an opportunity for every team

member to ask questions about the work or its context, ensuring a clear understanding of the task at hand. The facilitator may also want to discuss the exit criteria and clarify the expected outcome of the review. Lastly, the review process itself will be discussed. This discussion aims to ensure that all participants are clear on the methodology, the responsibilities of each role, and how issues should be documented and communicated.

Individual Review

During the individual review, participants work independently, adhering to established procedures, rules, and checklists to identify and document defects. Reviewers are tasked with noting defects, recommendations, questions, and comments. All identified issues are typically processed and logged. To ensure a thorough examination, reviewers or checklists may be assigned a specific perspective, as some defects become more apparent or severe when viewed through the lens of a user or maintainer.

The duration of the review is determined based on prior discussions. For instance, a formal review of 1-2 pages might allocate one hour of review time per page. This guideline should be strictly adhered to in order to avoid excessive time spent during preparation

Issue Communication and analysis

Issues identified during the review are communicated to the author or the person responsible for rectifying defects. This communication can take place through conversations, email, or during a review meeting.

Logging During the Meeting: All defects identified in the individual review phase are logged, and a severity grade may be assigned to each. The severity, suggested by the reviewer or moderator, is recorded without immediate discussion. The initial focus is on documenting all defects to ensure a comprehensive understanding of the issues before deliberating on their significance.

Discussion During the Meeting: Once logging is complete, participants share their opinions on the logged issues. Should discussions become personal or heated, the facilitator intervenes to maintain a constructive and respectful environment. The facilitator's role is crucial in ensuring that the meeting proceeds efficiently, with outcomes being either resolutions or actions designated for unresolved items.

Decision-Making in the Review: In the decision-making phase, the team deliberates on the logged issues to determine appropriate actions, such as corrections or improvements. If the product fails to meet the exit criteria due to significant defects, a plan for necessary rework is established. The reworked product is then subject to another review to verify that all concerns have been properly addressed and that it now complies with the exit criteria. This systematic approach guarantees the product's quality and adherence to standards prior to final acceptance.

Fixing and reporting

In the fixing and reporting stage of a formal review, the primary objective is to address the findings effectively. This begins with the creation of detailed defect reports for each identified issue that requires changes, ensuring a clear understanding of necessary corrections. Subsequent to report creation, the focus shifts to the actual fixing of defects within the work product, a critical step towards enhancing its quality and compliance. It's important to keep everyone informed about what defects are being fixed and who is responsible for them. Furthermore, the process involves recording of the updated status of each defect, potentially incorporating feedback from the originator of the comment to ensure comprehensive resolution. For more formal review types, gathering metrics, such as the number of defects fixed, is integral to evaluating the review process's effectiveness and improving future iterations. Finally, if the product meets all the required standards (exit criteria), it's considered acceptable. This whole process helps improve the product and ensures that everyone involved is accountable for making it better.

When working with writing a review no matter the scale, it is important to consider multiple factors to achieve a successful performance of a review. Some of these factors are preparation of the review writing, where it is important to have a clear and collective objective and review scope, and to have chosen the right type of review and techniques to use. All review material should be kept up to date including a policy that reflects this. In addition, all review personnel should have appropriate access to all necessary review material like documents, codebases and specifications. Sufficient training should also be carried out in relation to the review process, including techniques, tools, standards and guidelines being used.

Under the review process the reviewers chosen should do their work well. There should be a limited review scope where the most important parts are included. The participating parts should follow the review plan and possible checklists or roles methodically with regular managed review meetings in a trusting environment. Defects should also be identified, welcomed and expressed objectively. So that possible errors, coding bugs and security weaknesses are uncovered and corrected. During all feedback between any parts, there should be a consideration for the level of understanding of the receiving end. This way, all parts should be able to communicate well and understand each other. Transparency and objectivity are also highly crucial to ensure that reviewers and developers can openly discuss findings and concerns without the fear of the effects of bias.

One factor that also could help with the performance of a review, is how early the writing of a test review is started compared to the early stages of the development process. One of the testing principles followed when using reviews in the early stages of the development process is what we call *Early Testing*. This principle emphasizes that testing activities should start as early as possible and should be focused on defined objectives. This is in general to save time and money. For example, when identifying defects, it's much cheaper and easier to fix these earlier than at a later stage in the development. This also minimizes the risk of accumulating technical debt. Including that it also makes it easier to proactively find problems with the design or implementation before the code is completed.

Types and roles

There are 4 primary types of reviews varying in grades of formality. Ranging from least formal to most formal we have: **Informal review**, **Walkthrough**, **Technical review** and **Inspection.** For any type of review it's possible for management to step in to plan and allocate resources. The degree of which management contributes to a review will increase the more formal the review is. Management will plan when reviews should be done, who should participate in them, how often they should be done, etc. Each review can also have a scribe which will document the process of the review. This role is commonly combined with some other role (for example the review leader, author or a reviewer can do this job during the review).

Informal review

Starting with the informal review, this sort of review involves little to no planning or documentation of the review process. Examples of this can be pair programming or any sort of fast unorderly code review. This sort of review gives you some sort of benefit for little cost and is easy and fast, but is not sufficient by itself.

Informal reviews are usually just between two or more developers. The responsibilities here are to facilitate learning and understanding of the code between the developers. For the author this means explaining why they've taken the decisions they have and asking any questions that can help them improve the code. For the reviewer(s) this means pointing out any shortcomings, giving positive feedback where applicable and generally being a facilitator for the author.

Walkthroughs

Next in formality are walkthroughs. These sorts of reviews are a little more involved than an informal review. Here you can include people outside of the core development team such as stakeholders or management. Walkthroughs have the intended effect of letting developers share

their understanding of the software with both each other and the other non technical participants. In practice the formality and level of documentation for this level of review may vary, often dependent on who is involved in the review. These reviews can be applied semi-regularly to ensure that everyone in the project is on the same page and that no one has large knowledge gaps in important areas.

Walkthroughs involve having the code author lead a review where they explain the code to other developers and potentially some management and stakeholders. The role of the author here is to explain in non technical terms how the system works and listen to the feedback from the others. For the reviewer(s) their role is to learn the system, asking questions where they are unsure, giving feedback and potentially discovering defects which would then be documented.

Technical reviews

Technical reviews are similar to walkthroughs in that they can have a varying grade of formality. They are ideally led by a trained facilitator or moderator as opposed to walkthroughs where the review is led by the code author. In these reviews the involved parties are supposed to discuss and then define and document their suggestions for how to move forward with the code, including alternative approaches. They can also be used to solve technical problems with the code and check its conformance to the specifications or standards in the project's design documents. To get the most benefit out of such reviews there is usually some level of preparation beforehand where the review leader may prepare some material to be covered in the review. These reviews can be applied to let people share their opinions and to ensure the project is being developed in a way that allows the developers to make the best product.

In technical reviews the author will briefly describe their code and then enter into a discussion with the reviewer(s). In the discussion the author should have clear answers to why they’ve made the decisions they have and ask any final technical questions that are blocking development. The reviewer(s) here should voice their opinions on the matter and offer alternatives that can then be evaluated together to find the best solutions. They will also verify that the code is conformant with any specifications and standards, as well as documenting the discussion and its outcomes.

Inspections

The most formal type of review is an inspection. This is the final step before moving the project forward. The primary purpose here is to fix any sort of remaining defect. These are also led by a trained facilitator or a moderator. To pass a formal inspection there may be some exit criteria that need to be fulfilled, often in the form of checklists prepared by management. Since there is quite a bit of effort to host a formal inspection it’s vital that all entry criteria are fulfilled and all

involved parties are adequately prepared beforehand to not waste unnecessary time and resources on reviewing a project that is not yet ready for review. The product of such a review will include a detailed report of any findings, metrics, acceptable shortcomings, or a description of necessary changes to move forward. These reviews will often be applied near the end of deadlines for the project to assure it’s quality is sufficient to move the project forward or to ship the project to end users.

The responsibility of everyone involved in a formal review is to solve any remaining defects. To achieve this there are some different roles. The author will ensure the code has fulfilled all entry criteria and is ready for formal review. The reviewer(s) will examine the code beforehand and prepare any questions/suggestions for the author. In this review the documentation should include everything discovered during the review including whether or not the code is sufficient to pass review.

Static analysis vs. reviews

There are different methods used to test and analyze code before running the software in question, one of these methods is *static analysis*. Static analysis finds defects in software code and models. There are static analysis tools used to find these defects without executing the software. With reviews you can test software even before dynamic test execution. Both Static analysis and reviews are ways to test and identify defects early in the process without executing any software. Static analysis quickly identifies issues and enforces coding standards while reviews give deeper insight and more complex issues. In this way they complement each other and lead to a better quality assurance. Static analysis can highlight different areas with risk that reviewers should look at. The reviewers can also affect the static analysis by configuring the rules to fit the team's preferences and coding standards. The processes are different but together they give a more secure product and better testing then they could on their own.

Using reviews for quality assurance

Reviews vary in degree of formality and therefore can give different degrees of quality assurance. A more formal review will give better assurance but also use more time and resources. The degree of formality gets chosen based on factors like risk, size of the project and legal requirements. The given project therefore gets the right quality assurance needed early on in the process without using a lot of time. Since reviews can detect defects or lack of requirements early on in the process, it can stop bigger issues from emerging later on which will save a lot of time and money. Reviews promote continuous improvement, code consistency and knowledge sharing in the team that makes the software better and leads to less defects. Therefore reviews are an efficient means of quality assurance.

Reviewing this assignment

For the compulsory assignment we have chosen to follow an informal review type pattern that we found to be the most suitable given the severity and size of the different tasks. The different parts projected a low risk without any sizable need for planning. Including the absence of any needed legal requirements, apart from that we are not allowed to bring in others than the core team to work on the assignment.

In part 1 the informal review process helped us with getting a better understanding of the code. The team therefore could work effectively to complete the task, and the code has a better standard and structure because of it.

In part 2 the informal review process helped us with planning and mapping our expectations with the results we expected from the tests. We already had the knowledge of how to do unit tests and work with test-driven development, but had yet to perform it in this context and situation. For the TDD we followed the red, green, refactor steps suggested and felt we got good results from following this flow and reporting this to each other for verification.

In part 3 the informal review process helped us with exchanging knowledge of how the static analysis tools worked and how to solve the issues the analysis returned.

Conclusion

In conclusion, reviews have been described to be a valuable tool throughout the software development process, improving quality and communication among team members. Different types of reviews, ranging from informal to highly formal inspections, can be chosen depending

on the project's needs. Reviews can find defects early in the development cycle, which saves time and money compared to fixing them later. We feel that the use of an informal review process helped us communicate and work together better as a team. It also made it easier for us to verify that tests had passed as expected, defect detections, and that the different tasks were answered correctly.

Sources:

- Graham, D., Black, R., & Van Veenendaal, E., *Foundations of Software Testing*, 1st ed., Andover, Hampshire, Cengage Learning EMEA, 2019, pp. 75-86.