Assignment 1 - Group 24 - BDSA

base, chbl, luha

September 2022

Generics

The type constraints for the first method is that T must implement the interface IComparable. The type contraints for the second method is that T must inherit from U, with the condition that U must implement the interface IComparable.

Exercise 1

0.1 The noun/verb technique

1. ***Explain in which domain nouns and verbs that you identified are located***

   Description: I want a version control system that records changes to a file or set of files over time so that I can recall specific versions later. This system should work on any kind of files may they contain source code, configuration data, diagrams, binaries, etc.

   I want to use such a system to be able to revert selected files back to a previous state, revert the entire project back to a previous state, to compare changes over time, to see who last modified something that might be causing a problem, who introduced an issue and when, etc.

2. ***The implementation in libgit2sharp does neither contain a class File nor a class State. Explain how that can be when libgit2sharp***

Nouns
Version control system
File
Selected files
Previous state
Project
Issue
Version
Problem

Verbs
record changes
recall specific versions
work on any kind of files
revert selected files
revert entire project
compare changes
see who last modified something, that cause a problem
see who introduced an issue
see when something was modified

is an implementation of Git which is certainly a version control system as described above.

The Noun-Verb Method does not account for the analytical aspect (the application domain) and the design aspect (Solution domain) in the development process: *"File"* has to be represented explicitly. However, this depends entirely on whichever programming language or operating system it ultimately is implemented in. *"File"* could hypothetically consist of an implementation with interfaces and classes wherein *"File"* would be deemed futile/pointless.

Put concisely: The application domain and solution domain are 2 separate entities and thus however you present anything in each domain varies.

Exercise 2

Sommervilles types of Applications

The Coronapas App is mainly a transaction processing application. The reason is that the main function of the app is that a user wants to access their corona data. The app will then display the information so the user can understand the meaning. The argument for it not being a language processing application is that the user does not express their intention through a formal language.

Git is mainly a language processing application. That is because the user intention is mainly interpreted through formal language. The language can then be internally interpreted. The argument for it not being a transaction processing application is that the user can directly affect the database.

Exercise 3

Both Git and the Coronapas app are a *generic product.*

To consolidate the claim of Git's product type being a generic product, refer to the following paragraph:

"... However, the distinction between these system product types is becoming increasingly blurred. More and more systems are now being built with a generic product as a base, which is then adapted to suit the requirements of a customer."

Git is open-source and thus people can freely contribute to the product *and* distribute their *own* version of the product. It was developed with a generic product as a base, then people can adapt the software to however they desire, which then can make it customized software, be it for a customer or themselves. The Coronapas mobile app, however, is closed-source and distributed on the open market to any user who needs the app.

Exercise 4

Dependability, Security, Efficiency, and Maintainability

Coronapas App, Git, and the Insulin pump When thinking of different types of applications/products the product requirements vary.

Since a person can be dependable on an insulin pump the insulin pump has to be dependable as well. This is the main quality attribute that is strived for when making the insulin pump. Secondly, it has to be secure, so a third party can not tamper easily with e.g. the doses of insulin it provides.

Since Git is an open-source project its main quality focus is on maintainability. Its second priority is security, though it can be argued that this is a side-effect of being open-source.

The Coronapas Apps' main priority is to be secure since it operates with personal information.

This shows that they do not share the same quality priorities because of who and what the applications/products are made for.

Exercise 5

1. ***Explain why is there likely no architecture for Gitlet*** The Gitlet implementation is based on the predefined architecture from Git. The creator's intention was not to improve on Git and just learn how it works. So they ended up with a similar system as Git which only included all Git's most important features.

2. ***Git Architecture*** You could infer more about Git's architecture by thinking about who it was created for. Since it is created for programmers and

similar it has been prioritized to be maintainable, which is something that is thought about when creating a product and therefore it tells something about the architecture.

3. ***Difference between Git and Gitlet*** Since Gitlet is written in JavaScript and Git is mainly written in C, they have different restrictions. Therefore similar functions or functionality have to be implemented in different ways depending on the language. Gitlet focuses on a small subsection of Git's architecture, with only the most important features from Git.

4. ***Git and Gitlet's product characteristics*** Git prioritizes functionality and features to satisfy every programmer's version control needs. Instead, Gitlet focuses on essentialism, to implement the most important needs of the average programmer, for a version control system.

Exercise 6

1. ***Based on the articles, describe the reasons that caused the respective issues.***

   As stated in the article, one of the obvious reasons is the inadequate communication between the two parties: Developer and customer/user. There can be many underlying causes for this miscommunication, such as non-existent or inadequate documentation of the software for the customer, as not even the user knew about this so-called "unknown queue" intended for error handling, causing the very system itself to be esoteric and a black box for the intended customer.

   In regards to the wrong doses of medicine, an error occurred regarding the data integrity of a change in the code, causing people to get double their recommended doses, which is firstly caused - of course - by the lack of quality assurance of the software and - as described - a lack of IT experts.

2. ***Describe potential solutions to the problem.***

   As alluded to in the first answer, proper documentation and communication to the user (who presumably are oblivious to software development and regardless of the circumstances not their responsibility to know about the source code behind the software they use). Another solution to improve communication would be to rethink the error handling for the first case: If an "unknown" element within error handling exists, something is wrong. It should be transparent and explicit to the user that an error occurred, such as an red error message that appears whenever the error occurs, informing about the incorrect fulfillment.

   For the coding error, performing proper testing before publishing any coding changes should be of utmost importance when it concerns the health sector. Another solution could be to hire IT experts on the premises.

3. ***Compare your solutions to the proposed solutions in the articles, which one would you as a software engineer recommend? Argue for your recommendations.***

   Software should preferably not require formal training to utilize, especially when it concerns easily preventable errors such as lack of proper error messages: Even if you were trained in the software, it could be very possible that someone would forget and make the same mistake. In an industry where it depends on life or death, it should be absolutely clear if an error occurred. Therefore, formal training would be advisable in such a case as well of course, even if the system sufficiently explains itself and no side-effects of actions are present. However, even in the second case, I don't think anything should depend on an IT expert, as it is unreliable and should never be expected of a customer. We live in a modern age where the majority knows how to operate software to a lesser degree at least. Therefore, software should also be developed when the targeted audience are not experts in IT.

4. ***Discuss ethical dilemmas in case you were developing either of these healthcare systems.*** The overall ethical dilemma in making a health care system is based on a transaction between the programming firm and the healthcare firm. The healthcare firm wants to pay the minimum for the product and the programming firm wants to get the highest possible profit. Therefore the healthcare system may end up lacking. A possible solution may partly be to make an ongoing contract, to mitigate possible shortcomings.