AUTOMATED WEB APPLICATION FOR THE GATHERING, STORING AND VISUALISING OF OPEN SOURCE DATA.

Quality Plan

Abstract

Quality plan for the project of creating a web scraper that collects data from open source pdfs then collates, analyses and displays that data to improve users understanding.

Table of Contents

Introduction

Within the pharmaceutical industry, regulatory affairs departments work to keep track of several things; international legislation regarding the type of drugs their company is working on, ensuring the marketing and advertising of those drugs follow the requirements and restrictions posed by governing bodies and also keeping track of drug performance within the market. This project will be focussing on automating that final task, tracking drug performance. In this instance “drug performance” means how quickly a drug is approved for use once it has passed the final stages of testing. In order to track this, currently, an employee would be required to manually search for these dates on websites like the FDA (US Food and Drug Administration) and TGA (Australian Therapeutic Good Administration). They need to search for specific drugs that they are monitoring and find the date of approval, then they log this date so that comparisons can be made against their internal data to see how quickly they got to market compared to their competitors.

The aim of this project is to automate this process so that the resources in regulatory affairs can be distributed more efficiently by gathering, storing and visualising the external data for them.

Product Plans

There are three key elements to this project; data gathering, data storage, and data visualisation. These could be labelled as the three main milestones for the project and a fourth would be linking these aspects together to form one comprehensive program. Finally, an interface will be needed so that users can interact with the application as the three aspects would be created separately initially. Assessing these five milestones will generate a set of requirements and related outcomes. This section will include a high-level breakdown of the aforementioned milestones.

1. Gather open source data regarding drug inclusion dates.
    a. Web scraper visits official websites and searches for specific drug information.
        i. Decides on what constitutes an appropriate source.
    b. Extract text from PDFs when necessary.
    c. Find drug inclusion/approval date.
    d. Store information in a database.
    e. Automatic searches once a week.
        i. An appropriate time for the target website – based on time zone and work week.
    f. Users can force a search through the GUI.
2. Create a database to store data for the application.
    a. Store data from the data gathering portion of the application.
        i. Store company, drug information, and inclusion date data.
        ii. Store verified websites as sources.
    b. Store user data.
        i. Allow users to log into the application.
        ii. Store company affiliations and drug watchlist.
        iii. Affiliations with other users e.g. in the same company.
3. Create a dashboard that visualises the data in the database.
    a. Use charts and graphs to allow users to compare their company competitiveness in a certain market.
        i. Geographical/therapeutic.
    b. Export dashboard to report.
    c. Users can use filters to include/un-include data from the various charts and graphs.
4. Link the three components of the application.
    a. Web scraper can push data to the database.
    b. The dashboard can pull data from the database.
    c. Users can make changes to the database through the interface.
5. GUI – for user interaction with the application.
    a. Users can conduct a search for a specific drug or a specific drug in a specific country.
    b. Users can view the dashboard and filter information they are seeing.
    c. Users can log in to their account and make modifications.

The application development will follow an agile methodology inspired by SCRUM and structured following the V model, but with a few modifications, as it is a single-developer

project rather than a team. This will allow for the development to follow a feature sprint dev, test, release cycle with continuous testing which will suit the modular nature of the application. Additional steps like gathering user feedback can be added to the testing stage of the sprint when necessary, another benefit of developing in an agile way.

Process Description

In order to plan for quality, this section will identify the critical project processes and process quality standards that will be used to evaluate them [1]. Development processes will also be defined to ensure a consistent quality throughout the project lifecycle.

Development Process

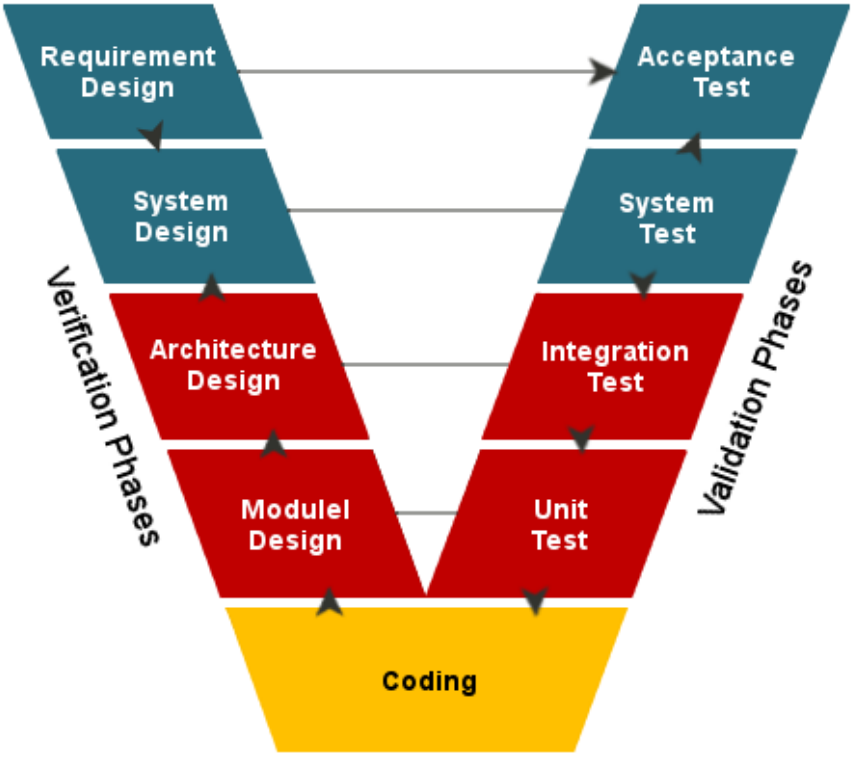

Figure 1: V Model diagram [2]

The V model in figure 1 accurately represents the development process of this project. At each stage of the project, there will be testing to ensure a high level of quality. The verification side works towards the analysis of the different aspects of the project and the validation side tests the project. Following this model throughout the project is key as "verification and validation is one of the software-engineering disciplines that help build quality into software" [3]. Evaluating the software during each phase using static testing ensures that requirements from the previous stage are met. Similarly, dynamic testing at each stage ensures that the product is meeting the requirements and is also producing the desired outcomes. As the model indicates, verification and validation will be present throughout the development process and due to this being a single person project these stages can be integrated seamlessly during each stage.

Component Lifecycle

This project will be managed in an agile way and the component lifecycle defined below will reflect this. Using a continuous delivery software strategy will allow the project to progress as efficiently as possible. Continuous delivery aims to “create a repeatable, reliable and incrementally improving process for taking software from constant to customer” [3]. The component lifecycle pipeline used in this project will include development, integration, and review stages. These will be done in a sprint-like manner inspired by SCRUM [4] and adapted for this single developer project.

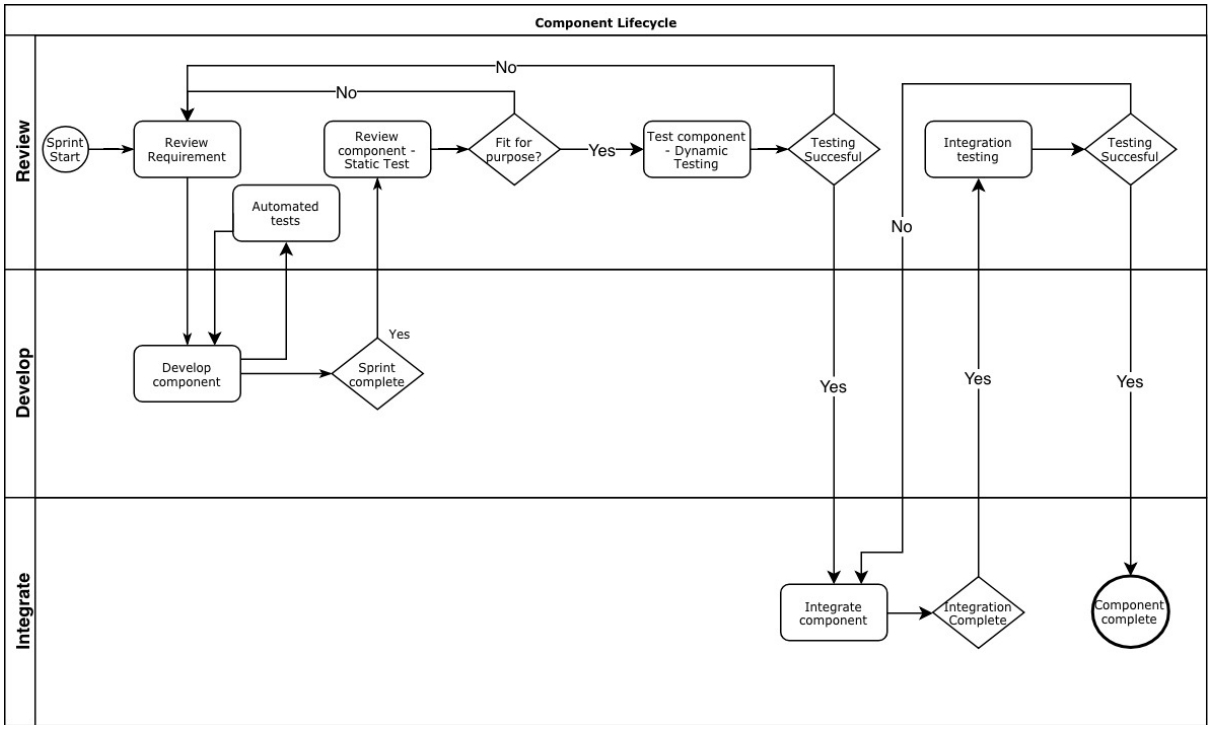

Figure 2: Component Lifecycle flow diagram

The process outlined in figure 2 will be followed during the development lifecycle of each component until they come together to form a key element of the project. Figure 2 shows a high-level overview of the component’s lifecycle including the three swim lanes that the necessary tasks fall under; review, develop and integrate. Once all key elements are completed the process will be followed for them treating them as components of the main application.

The first stage of the cycle would be to review the requirements. To ensure the component passes a quality review it is essential that the true purpose of the component is understood and the outcome for the sprint is outlined. This step should reduce the risk of unnecessary components being developed as well as needing to re-visit components later in the development cycle to re-work them to be fit for purpose. Developing each component in this way, following a SCRUM-like approach, will allow the program to be modular which will increase the efficiency in the review stages. By developing and reviewing the program in chunks it should allow for a more effective review process as we can review the component itself and then the main program once the new component has been integrated in order to ensure the process is going smoothly.

During the development stage, the component will be built over the course of the sprint. Once the sprint has ended the product will be reviewed against the initial requirements. If the component was completed and aligns with the requirement it can move on to be tested. Sometimes a feature is not completed in its designated sprint as there were unforeseen complications. When this happens, the feature will rollover to the next sprint which is reflected in the loop between feature review and requirements review. At this stage, a testing recursive loop could be added to ensure that continuous testing is undertaken whilst developing the component. This will be explored further in the v-model section of this quality plan.

If the development of the feature was completed during the designated sprint, then the component can be moved on to the review. The initial review of the component will be against the requirements to make sure the component is fit for purpose and is producing the desired outcomes. Once this static testing has been completed the feature can move along to dynamic testing at a component level. Once the component passes these tests successfully it will be integrated into the main application and undergo integration testing to ensure that the new component works well within the application itself as well as regression testing to ensure that the new code has not had a negative effect on older components. Once this testing has been successful the component lifecycle is complete.

Version Control

Throughout the project, GIT will be used to version control the code that is produced. Version control will "record changes to a file or set of files over time so that you can recall specific versions later" [6]. Developing components in this way should aid with integration. There will be a master branch that all working, integrated and tested components will be pushed to. A dev branch will be used for component development and integration testing to ensure that master will always have the latest, working copy. Merging to develop suggests the component has passed its unit testing and is ready for integration tests. As this is a single person project there will not be a need for separate branches for the development of each component as there is no risk of developers overwriting each other's work here. The dev branch will be sufficient in this project.

Continual Improvement

To ensure the project has a high level of quality a continual improvement process will be implemented. Evaluating the efficiency, effectiveness, and flexibility of project processes will allow for improvement. This will ensure that the project is meeting its quality goals.

Quality Goals

This section aims to quantify the "quality of application as well as quality of testing" [7]. ISO/IEC 25010 [8] identifies 8 areas that a product can be evaluated in order to determine its quality. There are two parts to ISO/IEC 25010; the first applies to the product quality and the 8 characteristics it defines, the second is the quality in use and the characteristics that are needed for a product to meet this.

Characteristics	Sub-Characteristics	Definition
Function Stability	Functional Completeness	The degree to which the set of functions covers all the specified tasks and user objectives.
	Functional Correctness	The degree to which the functions provides the correct results with the needed degree of precision.
	Functional Appropriateness	The degree to which the functions facilitate the accomplishment of specified tasks and objectives.
Performance Efficiency	Time-behaviour	The degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.
	Resource Utilization	The degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.
	Capacity	The degree to which the maximum limits of the product or system, parameter meet requirements.
Compatibility	Co-existence	The degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.
	Interoperability	The degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.
Usability	Appropriateness recognisability	The degree to which users can recognize whether a product or system is appropriate for their needs.
	Learnability	The degree to which a product or system enables the user to learn how to use it with effectiveness, efficiency in emergency situations.
	Operability	The degree to which a product or system is easy to operate, control and appropriate to use.
	User error protection	The degree to which a product or system protects users against making errors.
	User interface aesthetics	The degree to which a user interface enables pleasing and satisfying interaction for the user.
	Accessibility	The degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.
Reliability	Maturity	The degree to which a system, product or component meets needs for reliability under normal operation.
	Availability	The degree to which a product or system is operational and accessible when required for use.

	Fault tolerance	The degree to which a system, product or component operates as intended despite the presence of hardware or software faults.
	Recoverability	The degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.
Security	Confidentiality	The degree to which the prototype ensures that data are accessible only to those authorized to have access.
	Integrity	The degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.
	Non-repudiation	The degree to which actions or events can be proven to have taken place so that the events or actions cannot be repudiated later.
	Accountability	The degree to which the actions of an entity can be traced uniquely to the entity.
	Authenticity	The degree to which the identity of a subject or resource can be proved to be the one claimed.
Maintainability	Modularity	The degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.
	Reusability	The degree to which an asset can be used in more than one system, or in building other assets.
	Analysability	The degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.
	Modifiability	The degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.
	Testability	The degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.
Portability	Adaptability	The degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.
	Installability	The degree of effectiveness and efficiency in which a product or system can be successfully

		installed and/or uninstalled in a specified environment.
	Replaceability	The degree to which a product can replace another specified software product for the same purpose in the same environment.

Table 1: Product Quality - ISO/IEC [9]

Table 1 outlines, in detail, the characteristics that will be used to determine the quality of the application created during the project. The application will meet the requirements and deliver the desired outcome in order to be deemed functionally suitable. Timing will be key to this applications performance efficiency to ensure it does not become a problem for the website it is gathering data from, this will also affect the applications compatibility. Usability is also an important characteristic. The ease of use for the user will determine whether the system is an adequate replacement for their current processes. Due to the nature of the application, it's quality will be directly linked to security as accessibility and integrity are key. The development process has been designed to increase the maintainability of the application. Finally, the portability is key as adapting to different operational/usage environments is important for this programme. Functional testing will allow the whole system to be verified against these quality characteristics.

Non-Functional Requirements

Usability – User Interface & Accessibility

The application should be easy for users to interact with and achieve their desired result. The interactions needed should be self-explanatory so that the user can navigate the application un-aided. User stories will be created to outline typical system uses. Documentation outlining the uses of the system will be written so that users can refer to it if necessary.

Reliability

The application needs to be available and up to date when the user's login.

Security

The web application has to have high security as the data that users can input will be company confidential information (CCI). Due to this, it is essential that users cannot view other user's data. The security for the open source data does not have to be as extreme as anyone can find this data online.

Maintainability

The application will be developed as components and therefore will be modular by design. Designing the application in this way will increase how reusable the separate components are as they will work independently to the main application. The modifiability of the application will be high as it could be applied to several use cases by changing the data you are looking to gather e.g. crime rates, sports scores. The application will be developed in a way that includes continuous testing from a component to the system level, this should ensure high testability.

Metrics

Measuring and maintaining a high level of quality is essential. During this project key metrics will be measured to assess the quality.

- Security Metrics
  - Endpoint incidents – how many endpoints have experience security issues, such as being able to see other users' data, over a two-week period. The smaller this number is the better the quality.
- Application crash rate
  - "How many times the application fails divided by how many times it is used" [10]. As this is not a transactional application working with money the crash rate tolerance will be higher. Not losing any data when this happens is of more importance.
- Code
  - Weighted Micro Function Points (WMFP) is a software sizing tool using a parser to understand source code and derive code complexity to create a final effort score [11].
  - Halstead complexity measures module complexity in source code as well as computational complexity [12].
  - Cyclomatic complexity measures function complexity which is correlated to testing difficulty [13].
  - Code Coverage - a "measurement of the degree to which a test or testing suite checks the full extent of a program's functionality" [14].
- Customer Satisfaction
  - Net Promoter Score (NPS) shows how likely a user is to refer you to others. The higher the score the better.

Risk Management

"Risk management means risk containment and mitigation" [15]. This section of the report will identify potential risks associated with the project and create plans to mitigate the issues. Classifying the risk will help to prioritise them.

Risk	Details	Mitigation	Impact	Probability
Functionality	Unused features are developed/features that do not meet the requirements.	Static testing on the requirements for the project and detailed design created to limit the chance of this occurring.	2	2
Performance	Application run-time exceeds user expectation whilst searching for new data.	Use algorithms with best time case possible applicable to the problem.	5	3

Compatibility	The application consists of the 3 main components which could run as separate applications – these need to co-exist in order for the project to work.	Integration testing to ensure that the 3 components mesh harmoniously.	6	2
Usability	Users need to be able to filter data to be able to get the comparisons they need to make informed decisions.	Ensure necessary functionality is built into the systems so that users can do what they need to do as easily as possible.	6	1
Reliability	Data needs to be up to date and readily available for users when they use the system.	Ensure that data scraping occurs during times where the website load would be at its lowest to ensure that the scraping is done quickly and efficiently. Also, flag data that is current and allow users to force an update on a specific drug and country if needed.	8	4
Security	The authenticity of the data needs to be high so the sources the scraper targets need to be secure sources. Confidentiality is key as users will be able to input CCI and therefore it is essential that other users cannot access each other's data.	Some sort of logic will have to be built into the web scraper to verify the data sources. Use secure logins and permissions to ensure that people only see their own data.	9	6

Table 2: Identifying quality risks of the project

Bibliography

[1] "Stage 3: Plan the Project," University of Wisconsin, 1 February 2006. [Online]. Available: https://pma.doit.wisc.edu/plan/3-2/print.html. [Accessed 15 October 2018].

[2] "V Model," Professional QA, 06 September 2016. [Online]. Available: http://www.professionalqa.com/v-model. [Accessed 17 October 2018].

[3] D. R. Wallace and R. U. Fujii, "Software Verification and Validation: An Overview," *IEEE Software,* vol. 6, no. 3, pp. 10-17, May/Jun 1989.

[4] contributor, "The Continuous Delivery Pipeline — What it is and Why it's so Important in Developing Software," DevOps, 29 July 2014. [Online]. Available: https://devops.com/continuous-delivery-pipeline/. [Accessed 17 October 2018].

[5] K. S. &. J. Sutherland, "What is SCRUM?," SCRUM.org, 2018. [Online]. Available: https://www.scrum.org/resources/what-is-scrum. [Accessed 16 October 2018].

[6] S. C. a. B. Straub, "1.1 Getting Started - About Version Control," Git, 2014. [Online]. Available: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control. [Accessed 20 October 2018].

[7] Rutvi, "Quality Goals: Define and Measure Quality of your application," Infostretch, 22 November 2009. [Online]. Available: https://www.infostretch.com/blog/quality-goalsdefine-and-measure-quality-of-your-application/. [Accessed 21 October 2018].

[8] ISO/IEC, "ISO/IEC 25010:2011," ISO, 2017 (Last Review). [Online]. Available: https://www.iso.org/standard/35733.html. [Accessed 21 October 2018].

[9] ISO/ISE, "Product Quality - ISO/IEC 25010," ISO, 2017(Last Reviewed). [Online]. Available: https://edisciplinas.usp.br/pluginfile.php/294901/mod_resource/content/1/ISO%25010%20-%20Quality%20Model.pdf. [Accessed 21 October 2018].

[10] S. A. Lowe, "9 metrics that can make a difference to today's software development teams," TechBeacon, 6 June 2016. [Online]. Available: https://techbeacon.com/9-metrics-can-make-difference-todays-software-development-teams. [Accessed 22 October 2018].

[11] ProjectCodeMeter, "Weighted Micro Function Points," Project Code Meter, [Online]. Available: http://www.projectcodemeter.com/cost_estimation/help/GL_wmfp.htm. [Accessed 22 October 2018].

[12] IBM, "Halstead Metrics," IBM, [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSSHUF_8.0.2/com.ibm.rational.testrt.studio.doc/topics/csmhalstead.htm. [Accessed 22 October 2018].

[13] IBM, "V(g) or Cyclomatic Number," IBM, [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSSHUF_8.0.2/com.ibm.rational.testrt.studio.doc/topics/csmcyclomatic.htm. [Accessed 22 October 2018].

[14] Sealights, "Code Coverage Metrics," Sealights, [Online]. Available: https://www.sealights.io/test-metrics/code-coverage-metrics/. [Accessed 22 October 2018].

[15] Casr, “Understanding Risk Management in Software Development,” Cast, [Online]. Available: https://www.castsoftware.com/research-labs/risk-management-in-software-development-and-software-engineering-projects. [Accessed 22 October 2018].

Appendix

Table of Figures

Table of Tables