Group members info

Fullname	email
Burkman, Brendan	bburkman@depaul.edu
Chaidez, Arturo	achaide3@depaul.edu
Peraza, Eliecer	eperazaa@depaul.edu

Project Specifications

Context

Software quality is defined as requirements that software must comply with. Some of these requirements are explicit, such as standards and characteristics that must be met. There is a wide range of requirements: no crashes, defined features working properly, no unexpected results when using said feature and more. The requirements may differ from one software to another, but the software still must meet its own defined requirements.

Other requirements may not be specifically stated. For example, ease of use of the software and the readability of the code. Certain requirements may not be very well defined. It may be that the software is expected to be efficient in how it uses its available resources, but there is nothing stated that there is an unacceptable threshold.

As stated before, the most common and obvious requirements are the features of the software function and present no unexpected results. What may not be obvious is, as a software expands, does it affect previous code and makes it so it no longer meets initial requirements? How do developers deal with and prevent these bugs? In this project, we will build a calculator program from scratch, adding features such as arithmetic operations, unit conversions, and more over time, to see if we run into these problems and see how we fix them while applying techniques to assess and ensure software quality and proper testing.

Motivations/Problem Statement

In our professional field, we face software programs that do not comply with the minimum requirements to be cataloged as decent. Code that includes multiple bugs, no readability, lack of encapsulation, bad design, no documentation, no reliability, impossible to maintain, lack of extensibility or standards, and so on. As a result, multiple individuals/users have to deal with applications, machines, websites, and other devices that do not work properly or have a very low performance. Then, why not start adopting and using techniques, strategies, and tools that allow programmers to comply with the minimum requirements of good codes from the beginning, to have a fully, efficiently, and properly working product in the end?

As we design and develop this product, techniques of good design will be learned, applied, and integrated into this project. As the project progresses, the team will be able to use

quality assurance software and analyze the results it provides, and convert those results into useful reports. In the meantime, software testing will be performed on all parts and modules of the project, including individual and integrated testing.

We want to develop a calculator because we determined that it will be simple enough to be achieved in the given time, but still gives us a challenge while focusing on what matters for the goal of this course: software testing and quality assurance. We are going to implement the different features of the calculator using design patterns, quality metrics, software tests, and any other tool/approach we deem necessary to achieve a product of great quality and error-free.

Expected contributions(Solution) and Objectives

CalcPlus will allow the user to perform the basic arithmetic operations, as well as unit conversion in an intuitive, effective, efficient and easy manner. Each feature will be added incrementally, with its own set of JUnit tests (junit.org). The tests will not only be considered for individual classes and modules but also the classes and modules will be combined and tested progressively until the entire system has been properly integrated and tested as a whole. Operations and other features being taken into consideration for this development are listed below:

Basic Calc functionalities

- Addition, Subtraction, Multiplication, Division
- Negative numbers, Decimals, Percentage, Square root, etc.

Basic Unit Conversions

- Area, Length, Mass, Temperature, Digital Storage, Frequency, Time, and so on.

Potential Future Functionalities

- Basic Mortgage Calculator.

The calculator will also provide a graphical interface.

The main goal of this project, beyond its functionalities and the easy and practical use of the final product, is the design and development of a product with good quality (quality assurance) and bugproof (software testing), giving us the opportunity of applying the knowledge being gained during this course.

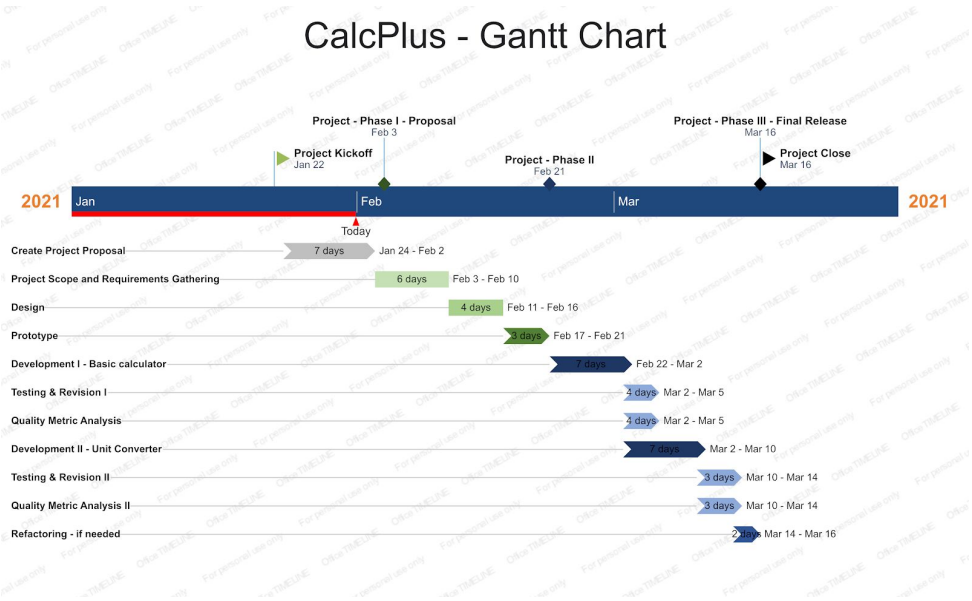
As each feature is added, JUnit tests will be created for the feature. Previous tests will be re-run to ensure previous code continues to work as expected, along with new tests to ensure new features work alongside old features. The purpose of creating features incrementally is to see if new code breaks previous code and if it does, document what went wrong to prevent the same bugs going forward. This way, later features will be designed with these possible issues in mind.

The quality of the product will be ensured based on the analysis of metrics obtained with Understand (scitools.com), which will be eventually reported, as well as incremental tests throughout the entire development to provide a bug-free, properly working product as the final result. Reports of analyzed data from Understand and JUnit tests might be provided on some phases of the development process.

Tasks management

For purposes of task management, the team will use the functionalities provided by google docs to list and assign tasks and responsibilities along with a very high-level Gantt chart

and high-level project activity list, to have available an overview of the project schedule and progress. Documentation of the JUnit test results, along with bugs that arose will be available, as well as charts of total failures and bugs, if applicable. Additionally, the team will have available a brief analysis of some metrics provided by Understand.



Title	T/M	Start	End		%
Project Kickoff	M	01/22/2021	01/22/2021	-	100
Create Project Proposal	T	01/24/2021	02/02/2021	7 days	95
Project - Phase I - Proposal	M	02/03/2021	02/03/2021	-	95
Project Scope and Requirements Gathering	T	02/03/2021	02/10/2021	6 days	%
Design	T	02/11/2021	02/16/2021	4 days	%
Prototype	T	02/17/2021	02/21/2021	3 days	%
Project - Phase II	M	02/21/2021	02/21/2021	-	%
Development I - Basic calculator	T	02/22/2021	03/02/2021	7 days	%
Testing & Revision I	T	03/02/2021	03/05/2021	4 days	%
Quality Metric Analysis	T	03/02/2021	03/05/2021	4 days	%
Development II - Unit Converter	T	03/02/2021	03/10/2021	7 days	%
Testing & Revision II	T	03/10/2021	03/14/2021	3 days	%
Quality Metric Analysis II	T	03/10/2021	03/14/2021	3 days	%
Refactoring - if needed	T	03/14/2021	03/16/2021	2 days	%
Project - Phase III - Final Release	M	03/16/2021	03/16/2021	-	%
Project Close	M	03/16/2021	03/16/2021	-	%

References

Duke, R. D. (n.d.). *Gannt.com*. Gannt.Com. Retrieved January 27, 2021, from http://www.gantt.com