CMPSC 280
Principles of Software Development
Fall 2015

Laboratory Assignment Four: Team-Based Creation and Presentation of Software

Introduction

Now that you know how to use a wide variety of software tools and you are more comfortable with eliciting requirements from a customer and then specifying, designing, implementing, validating, verifying, documenting, and releasing a programming systems product, we will continue these efforts for a real-world system that is clearly useful to other computer scientists and software engineers.

As you complete this assignment, please continue to pay careful attention to both the phases of the software development lifecycle and the roles of the members of a software development team, as explained in Sections 1.5 through 1.8 of the textbook. Additionally, all of your team members should carefully review the content at the end of Section 2.2 in SETP to learn more about the tasks associated with the validation and verification of software. Next, your team members should study the content in Section 3.1 so that they can learn more about project planning and the best way to define project milestones and activities. Using the knowledge from this content in SETP, your team members should all be able to outline the activities and milestones that are relevant for the completion of this assignment in one week. Finally, they can examine the content in Section 3.2 of SETP to learn more about the roles in a software engineering project and the working styles that team members may exhibit during the completion of this assignment.

It is also important for your team members to study Chapter 1 of MMM so that you can learn more about the effort that is necessary to implement a programming systems product — the ultimate goal that you should complete for this assignment. Finally, please examine the content in MMM's Chapter 2 so that you understand the different types of tasks that are normally completed during software development and how knowledge of a task's characteristics may better enable you to estimate completion times. Please see the instructor if you have questions about these readings.

Organizing Your Software Development Team

Please organize yourselves into teams of precisely three students. Whenever it is possible to do so, please make sure that you are working with individuals who were not members of your team in the second laboratory assignment. After reviewing the reading assignments mention in the previous section and discussing the software that you are invited to implement for this assignment, your team should discuss who will complete this project's tasks. As you are making task assignments, please think about the strengths and weaknesses of the members of your development team. For instance, you should ensure that, as best as is possible, you ask the "rational introverts" on your team to complete tasks that are most suited to their personalities. When it seems as though there are no team members who best fit certain roles, you should make compromises to ensure that all work will still be successfully finished. Finally, your team should ensure that all of your members know how to effectively use Git and Slack and then make a plan for how you will control your source code and documentation and communication using channel messages and integrations.

Eliciting Requirements from the Customer

For this laboratory assignment, your team is tasked with specifying, designing, implementing, documenting, and releasing a programming systems product that will give users insights into all of the Git projects in their filesystem. For the same of simplicity, this assignment sheet will refer to this product as `git-beagle` because it will "retrieve" the important details about all of the Git repositories that are stored in a user's directories; each team is encouraged to develop their own name for their programming systems product that best represents their tool's key features.

You should interact with the course instructor to learn more about the features that `git-beagle` should provide. Additionally, one of your team members should investigate existing tools or online discussion forums that outline the features that are common to software tools like `git-beagle`. However, the basic function of the tool will proceed in the following manner. When given a root file system directory, `git-beagle` will recursively traverse the filesystem and look for all of the directories that contain local or remote Git repositories. After finding a Git repository, `git-beagle` will collect and report relevant information about this repository. For instance, your tool might report files that have been modified but not yet committed and files that are currently in the directory but not tracked by Git. If `git-beagle` finds a Git repository that is "clean" and thus has nothing to commit, it should also report this information to the user. The `git-beagle` program should also report summary statistics that note details about, for instance, how many repositories were found overall and how many files were in the repositories but not yet added to Git's records.

For this laboratory assignment, all of the students responsible for requirements elicitation and analysis will interact with the customer at the same time. In addition, these students may assume, for this assignment, that the customer is a computer scientist who understands many of the intricacies associated with using the Git version control. When a team learns something important about the features of the system, then one of this team's members should share these details with all of the other team members through an appropriate channel in Slack. Since the ultimate goal for our course is to release a final version of `git-beagle` in a subsequent assignment, teams can share details about the system's requirements within the appropriate bounds of the Honor Code.

Designing the System

For this assignment, teams are encouraged to design and implement a system that uses whatever programming languages, development environments, and execution means that are most suited to the knowledge and skills of the team's members. In light of this freedom, the language used in this section and the following sections will be "generic" and thus not, for instance, refer to language concepts (e.g., "object-oriented" or "Java class") that are specific to a single programming language or development environment. Please see the course instructor if you have questions about this issue.

Working with the members of your team and leveraging the content in the requirements document, you should create a design for your system. As you are finalizing the system's design, you should try to develop answers to relevant questions such as: How many components will you use? What will be the relationship between the components? What functions will the components have? What will be the inputs and outputs of the functions? Is the design easy to understand? Can you actually implement and test this design? After answering these questions, you should use Markdown to write a design document with text and diagrams that explain the system. Since the main focus of this assignment is not design diagrams, students may use any reasonable tool to create their diagrams and, if necessary, include them as separate files in their team's repository. When clarification of the system's specification is necessary (e.g., due to writing ambiguities), the designers should interact with the individual(s) who elicited the requirements. Yet, your team should carefully ensure that the system's requirements and design are correctly in sync.

Implementing and Testing the System

The developer(s) and tester(s) on your team should take the requirements and design documents and start to think about how the system will be implemented. Your task is to ensure that the program faithfully adheres to the descriptions already produced by other members of your team. When an aspect of the requirements and design documents is not clear, the developer(s) must talk with the people who created these documents to resolve any outstanding concerns. The creators of these documents must quickly commit any changes in them to their repository so as to best ensure that the requirements and design of the system are in sync with its implementation.

As this programming systems product will be released to GitHub in a subsequent assignment, the implementors and testers should take care to create a system that is well-documented through comments and other relevant documentation. Whenever it is possible to do so, these members of your team must also add (semi-)automated methods for verifying that the implementation adheres to the requirements that previously were elicited from the customer. For instance, if your team decides to implement `git-beagle` in the Java programming language, then you should test it with automated tests cases that you wrote in JUnit, the industry standard for automated testing in Java. Overall, as you are implementing and testing your system you should hold yourselves to a high standard under the assumption that other software engineers will review and use your code.

Ensuring the Effective Operation of Your Team

When you start to work on this laboratory assignment, it may seem as though the designer(s), developer(s), and tester(s) "do not have any work to do" because the requirements of the system have not been established. Yet, if you carefully think about the work that you must complete for this assignment, it will become clear that this is not the case! For instance, one member of your team should be tasked with creating all of the needed means for communication with tools such as Slack and Bitbucket. Additionally, team members who are waiting to complete their chosen tasks should consider investigating the use of tools, like Trello, to organize their team's efforts. Finally, it is important for team members to spend time creating the templates for their deliverables and then carefully "sharpening their tools". For instance, the developer(s) on the team can ensure that they have a smoothly functioning development environment that will support the implementation of well-documented and correct code. Please see the instructor if you have questions about this matter.

Presenting Your Programming Systems Product

At the start of next week's laboratory session, each team will give a short five minute presentation explaining their own implementation of the `git-beagle` concept. You will be responsible for highlighting the key features of your tool and the ways in which you did your best to specify, design, implement, release, and document it. Whenever possible, you should give a high-quality, interactive, and engaging presentation that is both fun, interesting, and technically correct. Unless a team can demonstrate that it is not possible for them to implement their presentation as a program, all teams should use frameworks like `reveal.js`, `big`, or `beamer` to create their presentations. Creating your presentations using one of these tools is ideal because it will allow your team to store all of its content in a Git version control repository and thus ultimately be released with the rest of your programming systems product. Students who would like to learn more about implementing presentations with one of these frameworks can study some of the examples that are available in the course instructor's GitHub repositories available at `https://github.com/gkapfham/`.

Carefully Review the Honor Code

The Academic Honor Program that governs the entire academic program at Allegheny College is described in the Allegheny Academic Bulletin. The Honor Program applies to all work that is submitted for academic credit or to meet non-credit requirements for graduation at Allegheny College. This includes all work assigned for this class (e.g., examinations, laboratory assignments, and the final project). All students who have enrolled in the College will work under the Honor Program. Each student who has matriculated at the College has acknowledged the following pledge:

> I hereby recognize and pledge to fulfill my responsibilities, as defined in the Honor Code, and to maintain the integrity of both myself and the College community as a whole.

It is understood that an important part of the learning process in any course, and particularly one in computer science, derives from thoughtful discussions with teachers and fellow students. Such dialogue is encouraged. However, it is necessary to distinguish carefully between the student who discusses the principles underlying a problem with others and the student who produces assignments that are identical to, or merely variations on, someone else's work. While it is acceptable for students in this class to discuss their deliverables with their classmates, items that are nearly identical to the work of others will be taken as evidence of violating the Honor Code. In particular, make sure that your team members do not inappropriately share deliverables with the members of other teams.

Summary of the Required Deliverables

This assignment invites you to submit printed and signed versions of the following deliverables; please see the instructor if you have questions about any of these items. Please make sure that your team creates a version control repository in Bitbucket to store all of the deliverables for this assignment; since you will again select your own teams, please create a repository with a numerically-based name and share it with the course instructor. Also note that all of the written documents must be prepared with Markdown and, for submission, converted to PDF using `pandoc`.

1. A complete description of the roles that each team member fulfilled.
2. A requirements document that fully describes the features of your system.
3. Expressed in writing and as a simple technical diagram, the design of your system.
4. Well-documented source code that fulfills all of the system's requirements.
5. A tutorial that explains how to use all of the features of your finished system.
6. Individually completed and submitted reviews of all of the team members.
7. An informative and interesting five-minute presentation that highlights your software.