CSS 360 Autumn 2012 Project

Overview

One of my goals in CSS 360 is to provide you with experience on some of these fundamental tools, as they have profound and wide-ranging impacts on what it means to develop software systems. Reading about these tools is one thing. Using them provides a whole new level of knowledge and understanding.

Ideally, we would have enough time to work in teams to design, build, test, deploy, and evolve a fully functional application. However, 10 weeks is too short given the other material we need to cover from the textbook. Therefore, I have defined a particular project that will (a) provide you with some concrete experience with particular technical practices related to software engineering, and (b) give you experience working with a team to create the types of artifacts that manages need.

At times in this project I will act as your "manager", asking you to answer questions about your project, provide me with artifacts, or otherwise fulfill requests.

Description

The project that you will work on in your teams is to build an application like iBird. I will call our version BothellBirder. This is a mobile application for birders. Birding is an extremely popular hobby, supporting a multi-billion dollar industry, and there are already many players in this space.

Given the constraints of this course, we do not have time to implement an entire application like iBird. Instead, you will work in your teams to build specific parts of BothellBird that you can build, test, and deploy on a desktop. For instance, your team might focus on the "search" system, the data storage and retrieval system, managing the photos, etc. I expect that these subsystems will take 1-3 weeks to build each. You will be given some time in class to work on these.

One of the nice aspects of the application domain is that there are many examples of such mobile apps already, including:

 iBird

 Peterson Field Guide

 National Geographic Handheld Birds

 Audubon Birds

 Sibley eGuide to Birds

Most of these have free versions that you can download and use, and informative websites describing their product. This means that you students can use these free versions and websites to more easily and quickly understand the application domain including its stakeholders, uses, functional and nonfunctional requirements, types of data to support, and so on.

Professional tools and practices

Professional software development relies upon a suite of tools and technologies that enable effective software development in a team. These include Integrated Development Environments (IDEs) such as

eclipse and Visual Studio, pre-build libraries and packages including both commercial off-the-shelf (COTS) and open source software, and a variety of tools designed to help design, test, build, integrate, deploy, and evolve software systems.

Therefore, each person on each team will be responsible for using the following types of tools in order to understand the principles and practices involved:

- Tools:
    - **Programming language**. Examples include Java, Ruby, and C#.
    - **Source control system**. Examples include git, mercurial, bazaar, and svn.
    - **Continuous integration tool**. Examples include BuildHive, Bamboo, and CruiseControl.
    - **Integrated Development Environments (IDEs)**. Examples include Eclipse, NetBeans, and Visual Studio.
    - **Kanban system**. Examples include a physical board with sticky notes, LeanKit, and AgileZen.
    - **Issue tracking system**. Examples include Bugzilla, FogBugz, and JIRA.
    - **Unit testing framework**. There are unit testing frameworks for virtually all computer languages, and multiple frameworks for many languages.
- Required practices:
    - **Test driven development (TDD)**.
    - **Refactoring**. Many IDEs include refactoring support.
    - **Version Zero**. I will describe this in class.
    - **Testing**.

These documents will be graded based upon whether a sufficient range of choices were considered, the reasoning for the choice selected, and the clarity and professionalism[1] of the document.

Project management

Another aspect of developing professional software systems is providing visibility with respect to the state of the project. Managers and other stakeholders will want answers to questions lots of questions, such:

- How much will this project cost?
- What resources are needed for this project?
- When will this project be ready?
- How much has it cost so far?
- Does it fulfill our stakeholder needs? How do you we that?
- How will we know it is “done”?
- What are the risks and other dangers we need to mitigate?
- Is it secure?

[1] Note that diagrams that you include in your reports do not have to be computer-generated in order to be “professional”. Professional requires that you make good choices and use your time well. If a legible, hand-drawn diagram is easier to construct and modify, using it could illustrate a good use of your time.

- Will it be fast enough?
- How many users will the support? What will happen if we have too many users?
- Does it conform to our privacy policy?
- How will this system be maintained?
- How are you going to deploy the system to users?
- How can you effectively divide the work among a set of teams?
- How will you manage the work among the people on your team?
- At some point, you will leave this project. How can I be assured that a new software developer can effectively maintain and evolve your code once you leave this project?
- What are the important risks, and how can we mitigate those risks?
- What are you doing?
- Can you stop what you are working on now and add new feature XYZ?
- What can I do to help?
- Do you need any more people? If so, who?
- And on, and on, and on…

At certain times in the class, I will act in the role of "manager" and ask your team to answer some questions like those above. This may be done as homework, or as in class exercises.

Project reports. In addition to the above, I will want your team to provide me with a project update every week or two. These reports should be concise and effectively convey your project status. They will be graded based upon how well they convey your status, how well they answer questions around at the major risks on your project, and their clarity and professionalism.