Assessment of Prior Team's Work; Risks & Recommendations Document

Team Rigatoni

Created: 9/2/2014

Last Updated: 9/9/2014

# I. Summary of Recommended Actions

This section includes the subset of possible mitigation strategies listed in (III) that will be applied to our Senior Project.

* Port project to another architecture and remove the Eclipse RCP framework from architecture.
* Adopt a web-based framework for building the new project version. A web framework was suggested as an alternative to RCP by last year’s team. Alternatively, Node Webkit can be used to create desktop applications in Node.js if a web-based framework isn’t desirable.
* Apply an incremental delivery process. This best meets the communication needs of the sponsor, and it’s been recommended by the department and has the most likely outcome of project success.

# 

# II. Prior Work Q&A

This section provides a discussion into currently available artifacts necessary for making informed project decisions.

* Can the provided binary be executed in a variety of environments (OS X, Windows, Linux, etc.)?
  + OS X: The binary opens, but no Software Module / Tools for analysis are loaded. Projects can be made and text files imported.
* Can the development environment be configured for development on OS X and Windows?
  + Windows: Development environment can be configured, but the provided analysis plugins could not be loaded due to apparent code defects.
* What is the overall code quality for the prior team’s work?
  + In the Linguine module, concerns are adequately decoupled.
  + Error handling is poor.
  + The total amount of content in the plugin modules (every module except Linguine) is incredibly limited. Most of the utilities and plugins consist of only a single class or a few classes, and there are very few plugins with more than a few dozen lines of code. What code is already there could probably be reproduced very easily if necessary, and there wouldn’t be much loss if that code were to be scrapped.
  + Test coverage is incredibly sparse outside of the Linguine module. There are only 2 test classes within the 10 plugin and utility modules. It’s very difficult to know how effective the code within is without the tests to refer to.
  + Test coverage is also quite sparse within the Linguine module: 7 test classes and 2 mock object classes for the entirety of the project.

# 

# III. Risks & Recommendations

This section provides a list of high-level risks identified from assessing the work completed up to this point. Recommendations / mitigation strategies are listed for each risk.

1. **Configuration of the Eclipse Rich Client Platform (RCP).** The framework used by the prior team, Eclipse RCP, was difficult to configure and has a steep learning curve. No member of the current senior project team has experience using this specific version for Eclipse, and we have no subject matter experts for any version of the Eclipse platform. There is a severe lack of any kind of test strategy, making it very difficult to find out what areas of the program are and aren’t functioning properly. Additionally, a significant proportion of the previous team’s work focused on the areas of the application that are most closely coupled to RCP, and very little focused on areas of functionality that would be relevant to the sponsor goals for the project.
   1. Remove the Eclipse Platform from our stack. This would entail a significant amount of effort to refactor existing code, but would likely help mitigate the risk of encountering future issues due to unfamiliarity with the technology. From the prior team’s Technical report, “One of the biggest factors [affecting the success of the project was] the team’s poor understanding of Eclipse”.
   2. Switch to a different Eclipse version. Older versions may be more stable and have more documentation-related contributions from the community. However, this would not mitigate the risk of our unfamiliarity with the technology. Additionally, there are risks associated with utilizing a technology that is already outdated.
2. **Support for multiple client environments.** The sponsor identified during our kick-off meeting the inability of the prior team to run the application in a subset of the necessary environments. Our final deliverable must function on the machines used in the COLA classrooms, the SE Lab machines, and other hardware that will likely be replaced on a regular basis as set by ITS and Department standards.
   1. Consider a web application architecture. This significantly reduces the breadth of concerns for deployment. E.g., with a desktop application, every time ITS re-images academic machines, our application would have to be re-installed. If we can assume minimally that all lab machines will have a web browser and internet access, then the deployment strategy is centralized to a remote server.