TPS Report Start: 4/13/15 End: 4/19/15

GigaPan Education (CMU) Ruby on Rails website

Marissa Morgan and Iain MacQuarrie

Task Description	Assigned To	Time Est	Risk Est	Worked Time	% Complete	Code Reviewed
learn gigapan api	Marissa	8hrs	4	4hrs	45%	Yes
Study up on Ruby	Iain	16 hrs	3	15hrs	90%	Yes
Create login page	Iain	8hr	3	2hr	10%	No
Test out Rails Admin and ActiveAdmin gems	Marissa	3hr	2	1hr	33%	No
Database Design	Iain	2hr	3	1hr	50%	No
Finish Feasibility report	Marissa/Iain	8hr	2	6hr	75%	Yes

Design for Testability

Design for Testability are the design decisions made in order to effectively test software. Examples of design for testability is designing a class or method such that it can be detached from the rest of the software. This enables the developer to run tests on specific chunks of code with running the whole software. Design for testability also deals with designing code for logging and automation.
https://msdn.microsoft.com/en-us/magazine/dd263069.aspx
http://www.infoq.com/articles/Testability
https://www.prismnet.com/~wazmo/papers/design_for_testability.pdf
http://www.ibm.com/developerworks/rational/library/content/RationalEdge/nov02/Pettichord_TheRationalEdge_Nov2002.pdf
http://www.testability.de/Publikationen/CONQUEST02_jungmayr.pdf

A few practices that need to be followed to implement test-driven development:

1. Use mock objects. Mock objects are stub objects with predefined return values for every function call that may be given to the regular object. These are used to reduce dependencies so that each class can be individually tested.
2. Use a separate class for system calls to input and output. System calls that rely on specific files or data that may not be there can cause problems, so to reduce dependency on specific system calls, an I/O class is used, which allows for the use of mock objects in regards to I/O.

Test Driven Development

Test driven development is an approach to development where the test are written first, then the production code. The purpose is that have the developer think through the design or requirements before implementing. This also helps in writing the simplest code to make the test pass, helping eliminate bugs and boost reusability. Some of the common pitfalls of test driven development are writing trivial tests, not running the test often enough, and writing too large of tests.
http://agiledata.org/essays/tdd.html
http://c2.com/cgi/wiki?TestDrivenDevelopment
http://guide.agilealliance.org/guide/tdd.html
http://martinfowler.com/articles/is-tdd-dead/

A few practices to follow for test driven development are:

1. Write tests for new functionality before adding new functionality, then add code until the test passes. This is the Test-First Development portion of Test Driven Development, and one of the most important factors.

2. Write one test at a time, and write only enough code to pass all the tests. This ensures that whatever you write, you will know that any failed tests are occurring because of the code that was just written.
3. Refactor code after it passes to make it more efficient and test that code again. Doing the refactoring after the test has passed ensures you have the correct logic in place to pass the test, and reduces the chance of error.

Software Design Patterns

A software design pattern is a repeatable solution to a common problem in software design. They can not be directly transformed into code, but provide an overarching template for development. There are 23 patterns, known as the Gang of Four, that are considered the foundation for all patterns. These patterns can be broken down into three different categories: creational, structural, behavioral. Creational patterns deal with class instantiation. Structural patterns deal with the composition of class and object. Behavioral patterns are about the communication of objects. There is criticism that design patterns do not significantly differ from other abstractions.
http://sourcemaking.com/design_patterns
http://www.dofactory.com/net/design-patterns
http://www.tutorialspoint.com/design_pattern/design_pattern_overview.htm
http://www.developer.com/design/article.php/1474561/What-Are-Design-Patterns-and-Do-I-Need-Them.htm
http://www.martinfowler.com/articles/writingPatterns.htm

Practices to follow to implement Software Design Patterns:

1. Reference Design Patterns when designing a solution. Design Patterns exist to help give advice on how to best structure code to solve problems. Referencing Design Patterns first when designing systems can speed up development time.
2. Follow only the design patterns that make sense for a solution. If the design pattern is not geared towards the application being designed, there is no need to follow it. Only follow those design patterns that solve the problem at hand.