M24: Assignment #1

Due on Wednesday, February 18, 2015

Ben Mora

SwanTech

Contents

Team Organisation and Collaboration

Team Name

The team agreed unanimously to **SwanTech** as our team / company name.
We will use Java package name **com.swantech** in all Java classes.

Team Organisation

The M24 module 2015 has only 4 people attending, but 5 roles are identified in the lecture notes, so we have to split one role. The roles assigned are:

Customer Interface Manager	Simon Hewitt
Design Manager	David Tacey
Implementation Manager	Ifetayo Agunbiade
Test Manager	Mohamad Khaleqi
Planning and Quality Manager (PQM)	*split between team members:*
PQM - Team management	David Tacey
PQM - Version Control and Code Inspections	Simon Hewitt
PQM - Code Documentation	Ifetayo Agunbiade
PQM - Coded Quality	Mohamad Khaleqi

The team will meet formally each Monday at 15:00, in the Computer Science lab at Faraday 206, and will meet informally to check progress after the Wednesday lectures.
Ifetayo is investigating team time management tools that support work breakdown and delivery timelines (WBS and Gantt chart capabilities are needed). This will be an online tool allowing all team members to share and update progress.

Team Collaboration

The team will collaborate informally through a Facebook group, which has now been set up. Formal documents that are part of the development and delivery will be maintained in GitHub, chosen for its strong team collaboration capabilities, version control and multi-platform support.

Team Role : Customer Interface Manager

Simon Hewitt will be the Customer Interface Manager, as defined in the lecture slides "Team Roles and Group Work" From the lecture notes, the Customer Interface Manager is responsible for:

1. The team's relationship with it's customers
2. Resolving ambiguities in requirements specifications

To do this I propose the following actions and deliverables:

- Schedule regular meetings with the customers:- Ben Mora and Bob Laramee, to be agreed with them
- Create populate and manage a user requirements document
- Ensure non-functional requirements meet customer expectations
- Create and manage the assumptions and issues list in the project control structure
- Ensure the team has understanding of the requirements through formal requirements specification reviews
- And ensure team understands requirements by informal discussions at team meetings
- Review the designs to ensure they meet the requirements
- Review the test plans and test cases to ensure user requirements are tested and validated
- Responsible for managing each stage sign-off, ensuring the team deliverables meet the requirements and the customer accepts these deliverables as fit for purpose

As we will be following a RAD approach rather than a traditional waterfall, the requirements document will not be signed off and frozen, but will be the guide for each RAD development cycle. The requirements specification will be updated as necessary from feedback from RAD delivery and review. We believe that allowing the customer to see, use and give feedback on developing work ensures that we are on track to deliver what is really needed rather that what was written down (with the possible ambiguities that can arise from the complex process of documenting requirements). Furthermore, this RAD approach enables the development team to propose improvements and design alternatives that may deliver a better product.

We will investigate requirement documentation alternatives over the next 14 days. This can range from an Excel spreadsheet to high cost , high complexity commercial offerings designed for huge teams and multi year projects. We will be looking for small scale, preferably open source solutions.

Team Role : Design Manager

The team's design manager will be David Tacey. The responsibilities of this role are to ensure all team members contribute to the design process and that the design is completed to a high quality. The design manager is also in charge of clearly and effectively documenting the projects design. As it is assumed Java is the programming language to be used, a matching object-oriented design approach is to be taken. This design approach has a number of benefits including that both data and function class objects can be understood and modified without disrupting the entire system. It is also easier to follow as the interactions between objects can be clearly mapped. However to successfully implement an object-oriented design we must adhere to the principles of SOLID.

- **S**ingle Responsibility: This principle states that every class in the program should have a single responsibility. Therefore it must be entirely encapsulated by the class and only have one reason to change.
- **O**pen Close: This principle states that entities in the software should be open to extension via inheritance but closed to modification of their source code.
- **L**iskov Substitution: This principle outlines that derived classes must be substitutable with their base class. They don't alter the behavior of the base class.
- **I**nterface Segregation: This principle states that interfaces should only include methods required so that clients don't have to rely on methods they don't use.
- **D**ependency Inversion: This final principle states that high-level modules do not rely on low-level modules and both instead rely on abstraction. It also states that abstraction should not depend on details but details should depend on abstraction.

To follow this design process and ensure the design is of high quality and understood by all team members throughout the development I suggest the following actions:

- Prior to beginning any project create a thorough design plan with methods to be followed, standards to be met and logical design phases.
- Upon receiving the program specifications meet as a group to discuss how the project can be abstracted and encapsulated so that it fulfills object-oriented design criteria.
- Produce an initial high-level design of the system, fully documented using Unified Modeling Language with aspects such as class diagrams, use case diagrams and state diagrams.
- Finalize design as group with any clarifications from customer via customer interface manager.
- During implementation meet to review, evaluate and inspect the design and ensure it has been followed.
- Document any and all additions and corrections to design and circulate to team.

Team Role : Implementation Manager

This role is to be handled by Ifetayo Agunbiade. Based on careful requirements and specification analysis of a given task or project implementation of the project features can commence. The responsibilities tied to this role deals with the implementation and project management strategies the company applies to building products that meet the requirements of the client.

Responsibilities

- Make sure there is adequate information about the project specification and software requirements
- Engage all team members in designing a decisive plan for writing of the program code
- Make sure all team members adhere to the coding and documentation conventions [1]
- Engage every team member in coming up with technical solutions
- Engaging every team member in identifying specification implementation
- Coordinating the team, assigning tasks and also making sure each team member has sufficient knowledge of how to go about implementation
- Liaise with the customer interface manager to make sure implementation and features development are in-line with the scope requirement and time schedule
- Engage team members in the assignment of sprint and tracking progress
- Determination of the time for deliverables. Development of a work schedule, work breakdown, and determining a date for completion
- Handling risk associated with failure of task completion

Versioning

A versioning scheme starting from 0.1 and incremented to 1.0 for the first release of the software, where 1.0 indicate final completion/release of the software. Code versioning would be managed with Git. The first product release would take the version 1.0. Project software is to be held in a repository, each project repository would have 4 main branches

- Master Branch
- Testing Branch
- Development Branch
- Individual Developer Branch(es)

This would make it easy to identify development stages, apply bugs fixes without affecting already working code, separate development processes and code modularity by tasks, features and subsystems. Designing the repository in this way ensures that any change in the code follows a process and moves in one direction from features Development Stage to Testing Staging to Master (Production stage).

Master Branch

For the project the master branch represents the software in its production state containing the accepted features after it has gone through the development stages, technical reviews and integration testing. At every stage of the project implementation the master branch must always run and be void of defects or errors. The master branch is always to be stable and form a functional working software.

Testing Branch

This branch contains project software code that has gone through, and passed code reviews for conformance with the agreed features. Code in this branch is to be tested. At this stage all team members would review test codes, and apply changes as applicable. Once the code in this stage is approved it is merged to the master branch.

Development Branch

This branch contains code merged from the individual developer branch. Here the entire branch is checked for conformance with coding conventions, software development best practices and code documentation. Upon code reviews, this branch is pushed to the testing branch for further testing. For the code to advance from this stage all team members must review all codes, ensure adherence to coding conventions, best programming practices, and make sure the appropriate features have been implemented according to requirements.

Individual Branch

This branch may contain individual developer branch(es) of the project code. Each branch is to be cloned from the working Master branch (Production branch). Code push is done to the development staging branch for further review.

Development Process

Development paradigms like MVC and third party frameworks/libraries maybe used in-order to facilitate implementation. Adoption of such software development models or third party software would need to be used according to how they suit the given project. The software architecture and methodology to be followed in the implementation of the software are all subject to the agreed design specification. As with proper implementation, design comes first and implementation would be tailored according to the design approach. Development processes would take an iterative and incremental approach so as to accommodate any changes to the project requirements where frequent customer collaborations and input is seeked from time to time through the customer interface manager instead of having a closed implementation scheme where once the requirements are made available changes are difficult to make.

Deliverables Schedule

S/N	Week	Tasks	Deliverables
1	Week 1	Identify all tasks and features required. Work break down. Identify objectives and feasibilities. Determine an architecture of implementation. Project development.	Methods/tools for monitoring implementation progress. Establishment of an implementation model to be followed. Sprints assigned to individual team members. Repository set-up. Java and git training plan if necessary. Plan for next week.
2	Week 2	Iteration of tasks from the week before. Development may be redefined to accommodate changes in the requirements	Progress report. Supplementary specifications (if any). Assessment of specification changes to delivery time, team source code and documentation review. Bug fixes. Deployment of finished features/tasks to master branch. Push working code to test staging environment for testing. Plan for next week
3	Week 3	Iteration of tasks from the week before. Development may be redefined to accommodate changes in the requirements	Team source code and documentation review. Progress report and possible risk assessment. Bug fixes. Push working code to test staging environment for testing.
4	Week 4	Integration phase	Team source code and documentation review. Progress report and possible risk assessment. Bug fixes. Push working code to test staging environment for testing. Plan for final deployment.
5	Week 5	Integration phase	Integration of the development branch to the testing branch. Source code and documentation review. Final deployment and test plans. Project sign off.

Assumptions like the number of weeks to be spent on the implementation is subjective to change. It might be longer or fast tracked to meet delivery time.

Task and Risk Management

After design, specification and requirements analysis, work would be distributed amongst team members. Sprints would be set-up using visual studio online TFS (Team Foundation Server) and Gnatt chart project planner as applicable. Visual reporting would be made available to each team member so they can track development progress and identify risks in terms of tasks not being completed as scheduled. In software development there is always a possibility of failure of some type of process so it is important every team member has access to information that helps identify, assess, and monitor progress of each development phase or feature. TFS provides online tools to help us manage time, help in planning and timely resolution of risks.

Team Role : Test Manager

The test manager role will be *Mohamad Khaleqi* responsibility. The test manager roles is a task with responsibility for the test effort's[1] success. This role involves code quality and code testing. For test manager there are some roles:

- Understanding the testing by analyzing the project requirements.
- Organize the testing meeting.
- Develop the test plan.
- Arrange the software and hardware requirement for the test.
- Document all testing result and prepare a report.
- Communicate with team members.

In order to perform this role, I should:

- Prepare Unit testing e.g. JUnit[2]
- Prepare Abbot [3] testing for GUIs.
- Integration testing[4]There are two types of testing:
  - Button-up integration testing.
    begin with Unit testing and then moving to higher level combinations of units called modules or builds.
  - Top-Down integration testing.
    In this test, first higher level modules are tested and then low level module.

  In developing software process, usually Button-up integration test happens first and then Top-Down integration test. [2]
- Software testing (Testing and debugging)
- Black Box and white Box
- Alpha and Beta testing
- Regression testing
- Ensure the team has understanding of the testing methods and also terms in testing unit.

[1] Testing is the process of evaluating a system and the components to check if they are working correctly or not. Basically, it means to check the system to find errors, gaps, bugs and missing requirement or on the other words testing is a process of analysing a software to detect the difference between existing and required conditions.

[2] Testing will not catch every error in the program, because it cannot evaluate every execution path in any but the most trivial programs. The same is true for unit testing. Additionally, unit testing by definition only tests the functionality of the units themselves. Therefore, it will not catch integration errors or broader system-level errors (such as functions performed across multiple units, or non-functional test areas such as performance). Unit testing should be done in conjunction with other software testing activities, as they can only show the presence or absence of particular errors; they cannot prove a complete absence of errors. In order to guarantee correct behavior for every execution path and every possible input, and ensure the absence of errors, other techniques are required, namely the application of formal methods to proving that a software component has no unexpected behavior.

[3] A framework for unit functional testing in Java GUIs. More info: `http://abbot.sourceforge.net/doc/overview.shtml`

[4] In some books it is called as Component testing.

Team Role : Planning and Quality Manager

PQM - Team Management

The role of team management will be dealt with by David Tacey. To do this I will be the one to ensure minutes are taken at every meeting then formally written up and distributed to the all team members with clear outline of tasks to be completed. Another aspect of this role is to check that all deadlines are being met and to ensure the team works together to take action if work falls behind. As well as this I will deal with organizing meetings and notifying all members about aspects of the project through communication channels such as email and facebook. Monitoring the project using the tools TFS and gantt charts will give visual reports on progress made by each team member. Allocation of task sprints and risk management would be facilitated with the use of the said tools.

PQM - Version Control and Code Inspection

This part will be handled by Simon Hewitt. Version Control is managed by GitHub, so I will ensure the team understand GitHub and how to use it, that they properly use check-out and check-in. I will have control over final deliverable versions, and managing branch merging. Code inspection will be peer review across the group. I will investigate mark up tools. The MicroSoft Word document review tools are a powerful tool for team review, I will seek something similar for plain-text code files.

PQM - Code Documentation

This part will be handled by Ifetayo Agunbiade. Here all implementation code are checked that they are properly documented. Doxygen would be used in code documentation generation.

PQM - Code Quality

This part will be handled by Mohamad Khaleqi. In order to check the code quality in Java, First I will use Eclipse IDE because of automatic code indentation and auto formatting. Then I will make sure the code is following Bob's Concise Coding Conventions and also Doxygen. In addition to have better and tidy code, I will use CodePro Analytix [5] and PMD[6] which are both Eclipse plugins. They are both perfectly fine with any problem in the code and both are compatible with Java code. For code quality I will use LTFCE definition. The code over its lifetime will be read many more times than written. So, good code is:

- **L**egible: Only the code(Not comment) should clearly state the indent and reader should easily make sense of the code.
- **T**estable: The could will be in the way that unit tester can do its job.
- **F**lexible: Dependencies, both on other code in the code base and arbitrary implementation choices, should be minimized.
- **C**omplaint: The code itself should be correct and meet the system requirements.
- **E**conomical: The code should be responsible in terms of using system resources e.g. memory and CPU.

[5] CodePro Analytix is the premier Java software testing tool for Eclipse developers who are concerned about improving software quality and reducing developments costs and schedules. More info: `https://developers.google.com/java-dev-tools/codepro/doc/`

[6] PMD is a source code analyzer. It finds common programming flaws like unused variables, empty catch blocks, unnecessary object creation, and so forth.More info: `http://pmd.sourceforge.net/`

Schedule, Format and Quality

After searching for suitable document templates, we believe the attached document from the Swiss Federal Institute of Technology, Zurich, provides and excellent template. It is in MicroSoft Word format, so we will reformat into LaTeX, taking the opportunity to omit sections that are not relevant and ensure it fits our needs. This answers the question on format, and contains the necessary guidance on quality management as well. [7]

Initial Schedule of Work

When we receive the project requirements on 18/2/15, the initial actions are:

- Arrange a team meeting as soon as possible to review the document
- Set up and populate an open issues register, add all open questions to the register (Simon)
- Initial decomposition into work domains, arranged by team strengths as far as possible
- Create a first draft WBS [8] structure and assign names and target dates to each work package.
- Customer Interface Manager (SH) to arrange meeting(s) with Ben Mora to resolve all open questions.
- Test Manager (MK) to create outline test plan, showing what elements will be tested by which techniques, and setting up any necessary infrastructure.
- Design Manager (DT) to create high level design
- Implementation Manager (IA) to create high level implementation plan, and take ownership and manage the WBS schedule.

All of these actions to be completed by Monday 2 March 2015.

Environment

It is assumed that the project will be developed in Java and so Java version 7 or latest version would be used.
The team members use different desktop OS including Linux, OS X and Windows, so tools must support each of these. We have agreements or working assumptions or proposals for all identified software components:

Language	Java	Assumption for the project
IDE	Eclipse or NetBeans	Individual choice
Desktop	OS X, Windows, Linux	Individual Choice
JVM	Java version 7	Awaiting input
Source and version control	GitHub	Agreed
Desktop source and version control	none specified, default is Git command line	Individual choice
Documentation	LaTeX	works well with GIT and is a academic standard
Informal Collaboration	Facebook	De facto standard
Testing	JUnit	The most widely used Java testing framework, simple to adopt
GUI Testing	Abbot	Eclipse plugin to handle testing which JUnit cannot
Code Quality	CodePro Analytix and PMB	Eclipse plugins which aid coding quality
Test runner	To be decided	Nightly build and test sequence is desirable
Time Management	TFS and Gnatt chart	Gantt chart, WBS

[7] se.inf.ethz.ch/old/.../Project Plan wo QA, Transition.doc
[8] Work Breakdown Structure, a simple way to manage deliverables in a project (http://en.wikipedia.org/wiki/Work_breakdown_structure)

References

[1] R. S. Laramee, *Bob's Concise Coding Conventions*, July 19, 2013.

[2] I. Sommerville, "Software engineering," 2009.