**Project Report**

We have been assigned a project to produce a piece of software that will create all of the WBT (Work Breakdown Tree), PERT (Program Evaluation and Review Technique) and Gantt charts for a project from a common data input. The project must proceed in accordance with recognised software engineering principles, for example a recognised system development life cycle. This is to give us experience of real world software engineering that we will encounter in the work place.

**Background**

The software that is already available online that produce these charts are not sufficient for the following reasons:

* Many are expensive
* Many don’t run on multiple platforms
* Many only support one type of chart instead of being able to produce all three
* Much of the existing software doesn’t support updates of the plans once they are created, which are almost inevitable for any real-world project.

We are presented with a complex coursework in order to provide us with real world experience such as:

* Experiencing a project through start to end in a team
* Communicating efficiently as a team
* Assigning suitable tasks to each team member
* Planning and implementation of projects

**How we are going to do it**

We are expecting to employ an approach based on the Waterfall model as an answer to the system development life cycle. The Waterfall model follows a linear, top-down approach which is appropriate for most systems.

**Why Are We Basing Our SDLC on the Waterfall Model?**

The short time period we have been given to develop the project eliminates the need for more advanced system development cycles, such as the R.A.D. model; the short development period inhibits the ability to incorporate a rapid, cyclic model such as R.A.D. or Prototyping. A more flexible approach to the Waterfall Model ensures that depth and precision is maintained while the ability to make changes late in the development is still viable.

**Constraints**There are several factors limiting our project development, one of which is time. From start to finish, we have an estimated 7 months of development time, which incorporates all stages of project development. For a software product, this is a relatively short time span, especially for a small team.

We are also constrained by method of version control: we are only permitted to use GIT, as per instructions. Finally, we are constrained by the fact that we are individually inexperienced in functioning in a software development environment as a team.

**Programming Languages**

As a team we decided to utilise Java as our primary programming language, this being because every team member is fluent in both Java and Object Oriented Programming. On top of this the IDE supports accelerated graphical user interface development, which would give us more time to produce functional code. The standardised graphics libraries used in Java are well-known and so many will feel comfortable using GUIs produced by them. Another advantage of Java is the fact it is platform independent, which minimizes our coding overhead due to minimal code refactoring required.

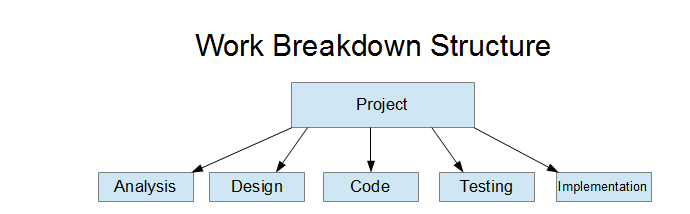
**Cloud Resources**

As a team, we have decided to use GitHub as our version control repository, as we are constrained by using Git. Using a cloud resource that natively supports Git version control is the logical option as it will minimise overhead with regards to translating to a repository that may not natively support Git. We also decided to use Git for many other reasons, such as its facilitation of simple yet effective collaboration between team members, its general developer-oriented focus, a minimalistic, utilitarian environment removed of bloat and clutter and the fact that it’s free to use.

**Diagrams**

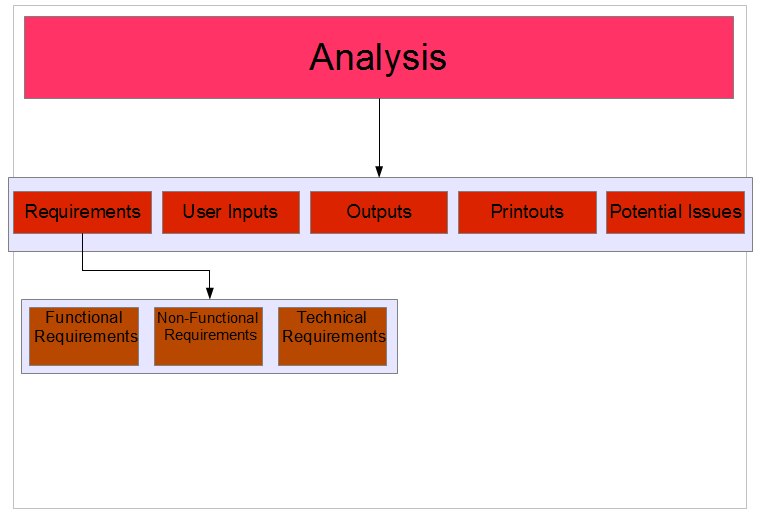
The following are diagrams illustrating our planning phase in a WBS, a PERT chart and a Gantt chart.

**WBS - Overview**



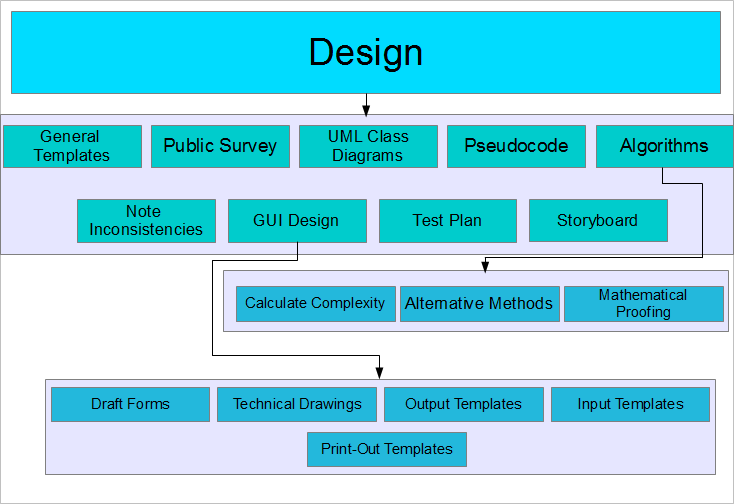
*This method of breaking down the workload follows tightly with the main principles behind the Waterfall SDLC.*

**WBS – Analysis**

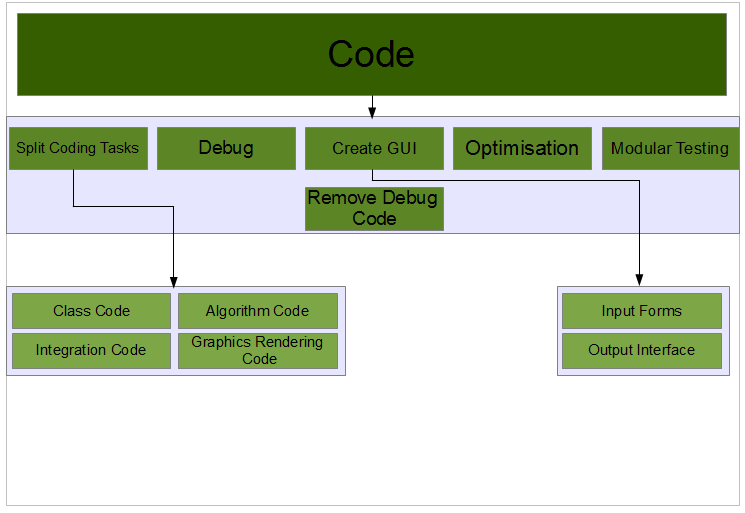
****

*Most of the tasks within the Analysis stage are self-explanatory and difficult to break down any further without taking a more convoluted approach, but the requirements was broken down into ‘Functional’, ‘Non-Functional’ and ‘Technical’ in order to streamline & segregate the requirements analysis process.*

**WBS – Design**

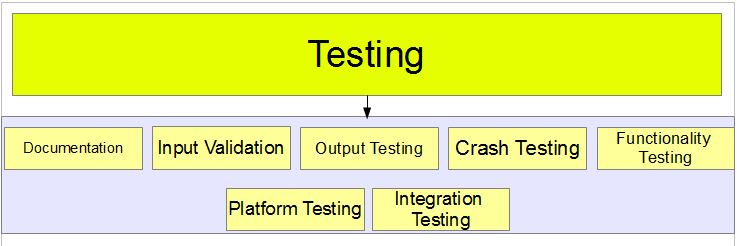
****

*Algorithm Design & GUI Design were broken down further into sub-tasks due to each task’s complexity. For many tasks, it is difficult to break them down further before the design phase actually begins.*

**WBS – Code**

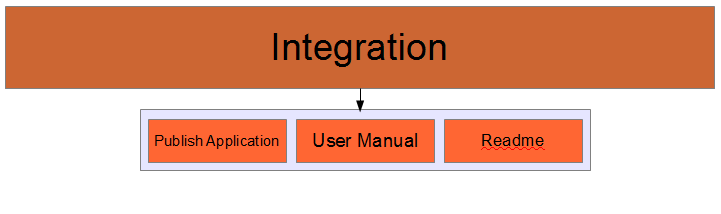
*Coding is split into several different, segregated tasks due to the nature of object-oriented programming. GUI is split into input & output, as these forms must be developed separately.*

**WBS – Testing**

****

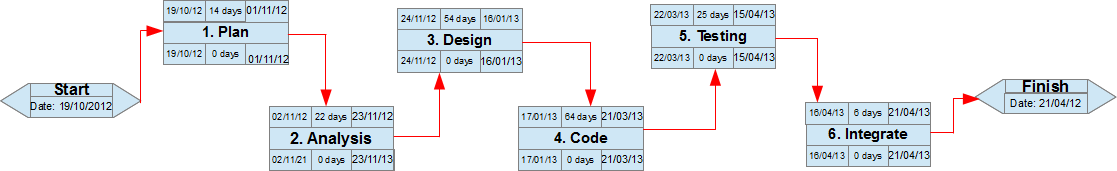
*The Testing phase is relatively self-explanatory. Since we have not entered the design phase, we cannot break it down any further at this stage.*

**WBS – Integration**

****

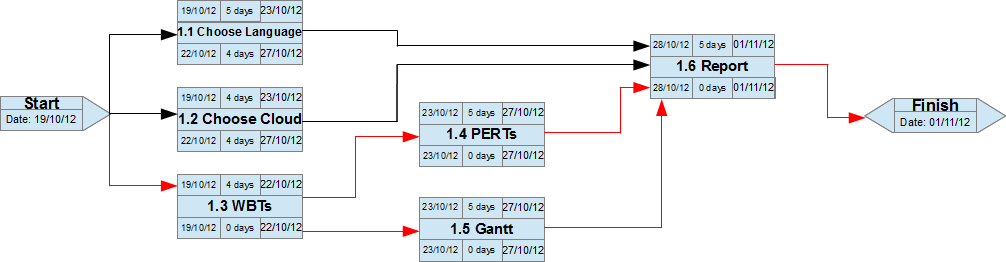
*We are expecting the integration phase to be relatively simplistic. The User Manual and the Readme will be written based on documentation, which is modified during the development process.*

**PERT – Overview**



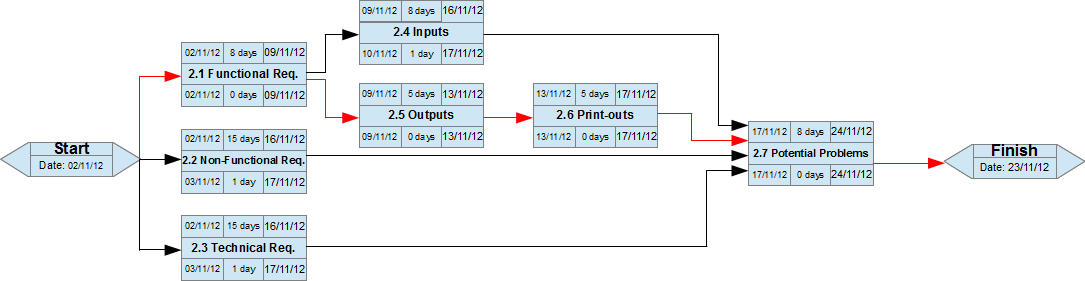
*The entire project follows a single critical path. The completion dates are relatively vague estimates at this stage.*

**PERT – Planning**

****

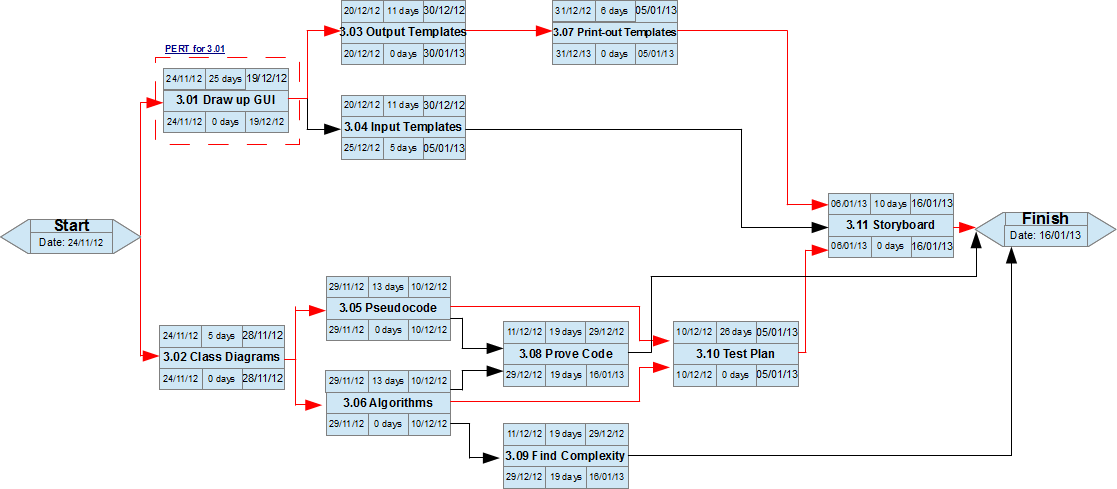
*The planning phase depends on the critical path following chart creation, since this is the most vital element of the planning phase and allows for no slack.*

**PERT – Analysis**

****

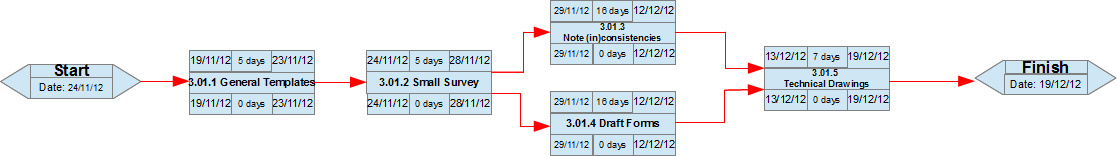
*As the PERT chart above clearly shows, the analysis stage is heavily dependent on the Outputs analysis task and its follow-up task, the Print-outs analysis.*

**PERT – Design**

****

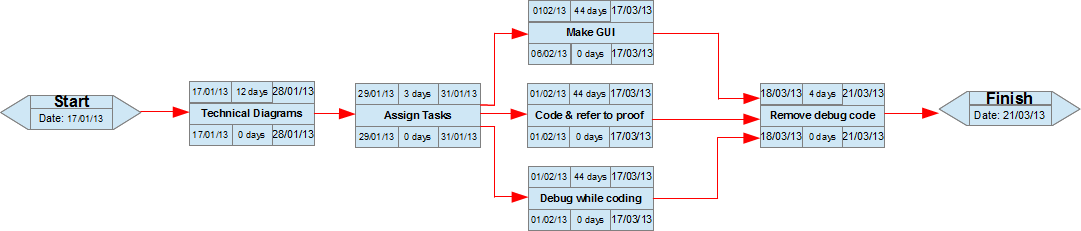
*The critical path is split into two main paths, as neither of these paths allow for slack.*

**PERT – Design 3.1**

****

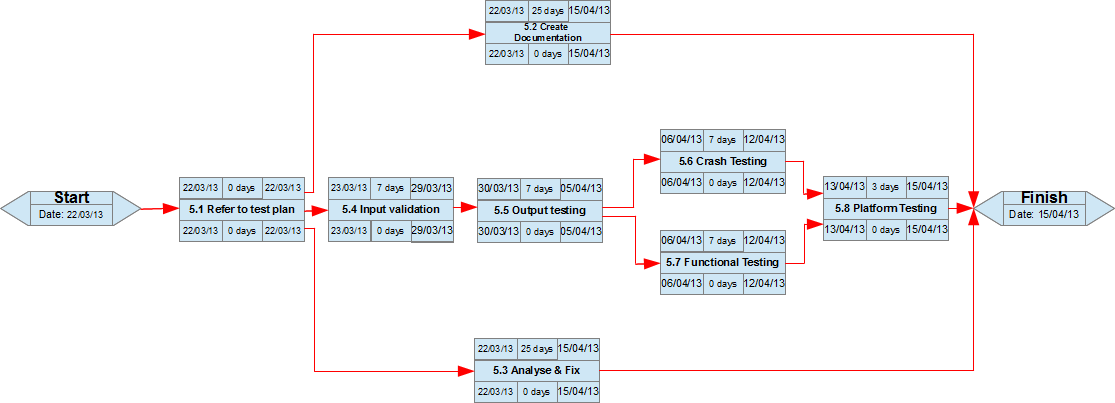
*Due to the tightness of the work schedule, no slack is permitted for drawing up the GUI.*

**PERT – Coding**



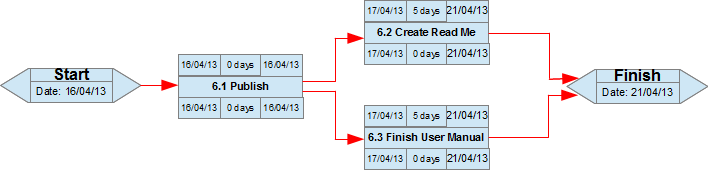
*Coding allows for no slack whatsoever according to our estimations.*

**PERT – Testing**

****

*The testing phase requires no slack whatsoever due to tasks having heavy dependency on one-another.*

**PERT – Integration**

****

*Whilst a simple task, integration also has no slack due to a predicted “close finish”.*

**Gantt Chart**

