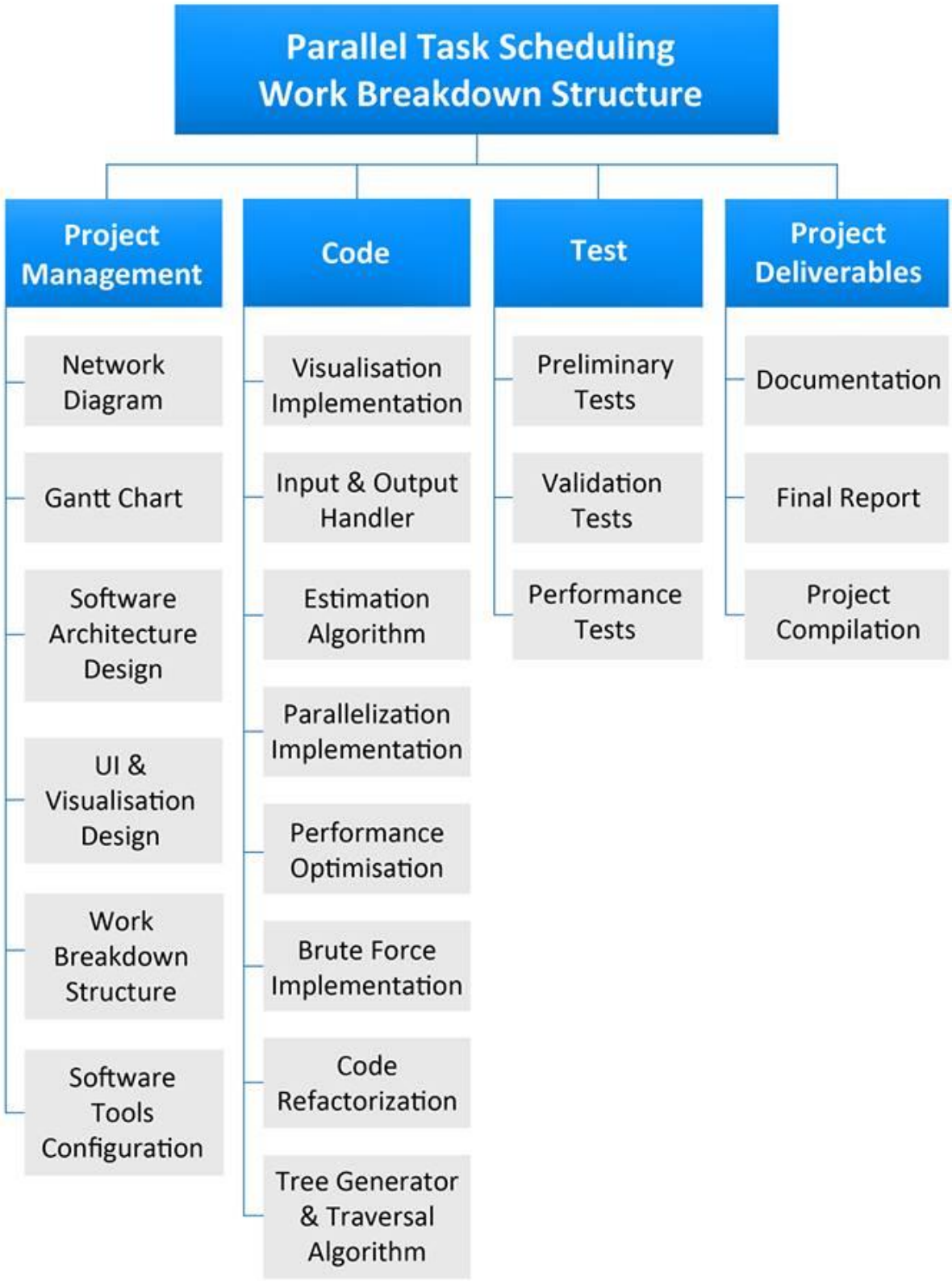
Project Plan

Jordan Wong, Ammar Bagasrawala, Ben Mitchell,
Hanzhi Wang, Kevin Yu, Jun Xu
PARALEX

Work Breakdown Structure



Outcome Descriptions

Work Breakdown Structure

The project is broken down into smaller sections in the form of a list of required outcomes. When the outcomes are put together, this should represent the entire completed project (100% rule). The work breakdown structure should assist in the planning and execution of the project since the large complex task is broken down into smaller and more manageable tasks. The list of outcomes should not be repeated to avoid doing redundant duplicate work.

Network Diagram

A Network Diagram will constructed based off of a previously created work breakdown structure (WBS). For each task in the WBS a duration will be estimated and assigned, dependencies and ordering of tasks will also be included. The process of constructing this Network Diagram will identify the critical path of building the implementation as well as tasks that have float times associated with them.

Gantt Chart

Based off the Network Diagram, each task will be visually represented on a timeline indicating how many days we will expect to spend on the task. Instead of the work hours estimate from the Network Diagram, the Gantt Chart shows a estimate in days, and serves as a loose project plan indicating preferred task start and finish dates that result in the project being completed on time.

Software Architecture Design

The team would have had a thorough discussion on the approach to the problem. The algorithm would be modularized with a description of the interaction between each module. This should be documented in the form of a class diagram so everyone understands how the modules work together. The main purpose of this outcome is to remove sequential dependencies so preceding parts of the algorithm do not have to be completed to allow their succeeding parts to be worked on.

UI and Visualization Design

This task entails outlining the information users would require in order to understand the processes taking place to parallelize the tasks. The design of the user interface would also be taken into account in this phase, which includes research on available tools and libraries and how data is to be represented to the users. Discussions would be held to differentiate between information which is essential for users and information which isn't, so that during implementation there exists a possible compromise.

Software Tools Configuration

All of the software tools that will be used to create the implementation will be decided on and configured appropriately. This would include but not be limited to, creating a private Git repository to handle version control, deciding on IDE to develop the implementation on, defining the project's coding practices and gathering and referencing external libraries and tools.

Visualization Implementation

The implementation of the visualization includes discussions between team members and the creation of several paper prototypes each of which are appropriately tailored for in terms of the chosen tools and libraries. Once a prototype is selected to be implemented, the basic design with the essential information is to be coded, and through iterations this base model will be enhanced in terms of aesthetics and amount of presented information.

Input/Output Handler

The module that will handle the raw input of the implementation as well as that which will produce the refined output. This section of the implementation will transform data inputted into the program to data structures that can then be used by further modules to identify the optimal solution. Once the solution is found this module will transform and output the data in an appropriate format.

Estimation Algorithm

The implementation of the module that will be calculating the estimated total cost if the current partial schedule were to be completed. The estimation has to be an underestimate from the true value to avoid removing a partial schedule that could potentially be the optimal schedule. Although the estimated cost will get closer to the true cost as the partial schedule becomes more complete, the estimation algorithm should provide reliable values even for 'early' partial schedules.

Parallelization Implementation

Once the scheduling algorithm has been implemented and tested for correctness, it will be modified to enable execution on multiple cores simultaneously. This will allow the scheduler to perform more efficiently on modern, multi-core devices by making more effective use of available resources. This will likely involve the use of an external library such as Pyjama or Parallel Task.

Performance Optimisation

Using results from performance tests, attempts will be made to mitigate the impact of performance bottlenecks on the overall run time of the scheduling process. The scheduling algorithm will also be improved upon by identifying which parameters and variants of the algorithm result in the best performance on sample inputs.

Brute Force Implementation

A brute-force approach to the problem will be implemented. This algorithm will be used mainly for creating test cases and guaranteeing the correctness of solutions created by our optimised solutions. While this will not be part of the final project deliverables, it will allow easier creation of test cases.

Code Refactorization

Refactoring the code is a task which would take place once a working prototype is completed. The existing code will be refactored to implement better software engineering practices. Code may also be refactored to increase performance in the performance optimisation phase itself. Even though the documentation indicates this task will take place near the end, it may be carried out at each milestone to reduce the amount of work at the end of the project.

Tree Generator and Traversal Algorithm

An algorithm will be created to generate a tree of possible schedules (and partial schedules) and traverse it to find an optimal solution. This will work together with the estimation algorithm to speed up the search using techniques such as branch-and-bound and A*. This module will be a major component of the scheduler program and will require interaction with many other components of the system.

Preliminary Tests

Preliminary tests are conducted before the parallelization implementation to test the correct output of our sequential solution. This stage will see team members going through multiple iterations of testing to ensure all the code compiles and works together. This stage is also critical as incorrect implementation can set back the whole project and expend unnecessary resources such as time.

Validation Tests
Validation tests are performed to identify errors in our visual and code implementation and ensures correctness in our solution. Correctness is administrated and enforced through our brute force implementation where any solution that is presented from our optimised solution would be congruent with our brute force solution. Multiple test cases, consisting of normal and extreme cases, would be used to cover the accuracy of our implementation.

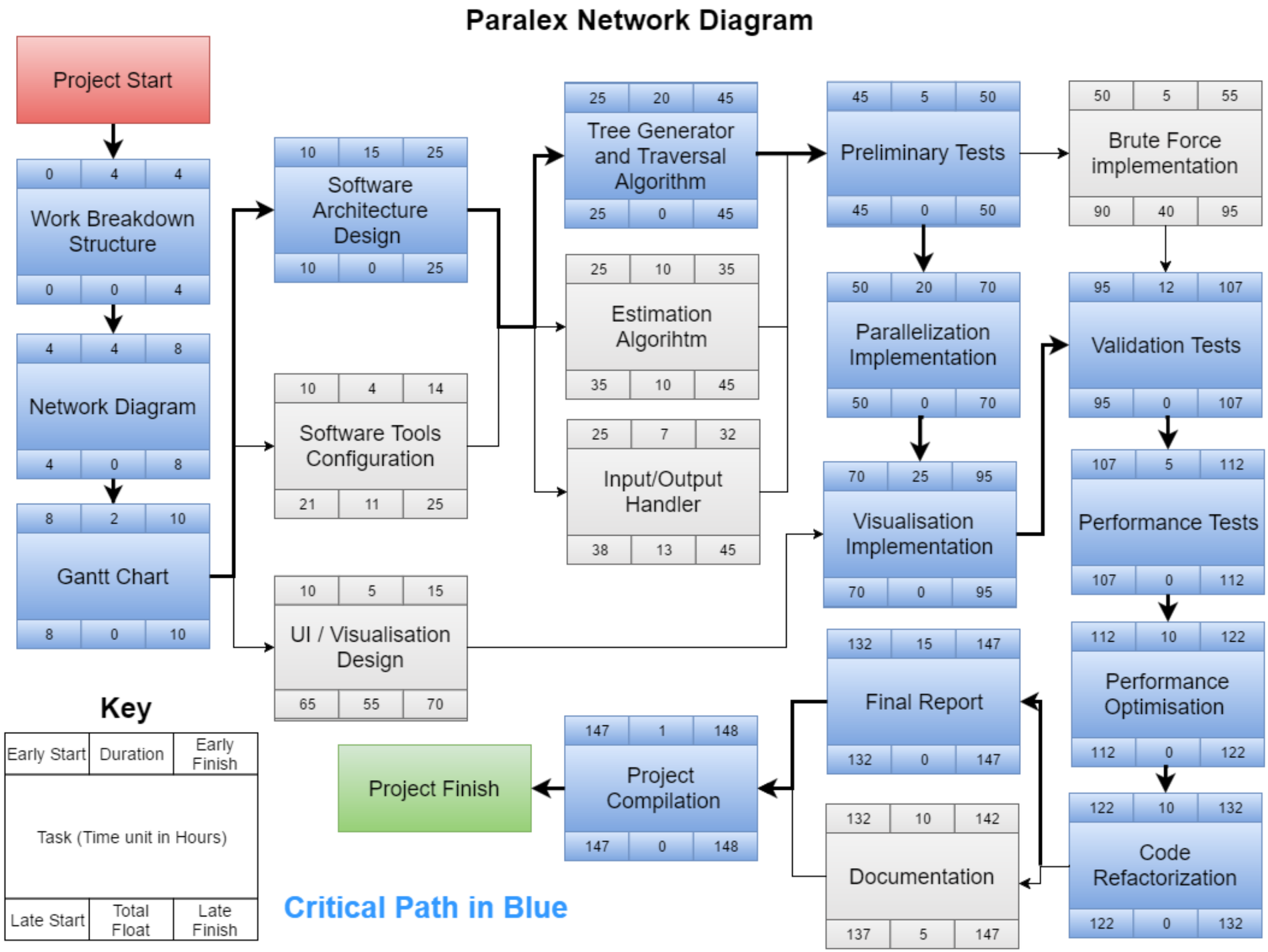
Performance Tests
Performance tests is used to measure the speed and efficiency of our code. The performance tests will be used to identify speed bottlenecks in our implementation such that improvements and optimisations can be carried out. Performance will also be measured against other potential implementations so that we get the most effective implementation. This phase would go hand in hand with optimisation to ensure a proficient solution is achieved.

Documentation
The outcomes of this is the uploaded files that highlights the discussions that the team has undergone so that everyone can refer to the appropriate files such as the Gantt chart of class diagrams when required. This also includes updating the GitHub wiki to describe the project plan and software engineering practices used. This mainly involves distribution of files rather than description of the files and methods used.

Final Report
The written report will contain a summary of our team's software development process and how we solved the problem. The report will go into detail about how we worked as a team and distributed the workload. We will also discuss unexpected obstacles we faced, how we dealt with them, and what we will do next time to prevent them from happening.

Project Compilation
At this point the implementation will return the optimal schedule and allow for parallelization and visualization options.,The completed implementation of the project will then be packaged along with it's dependent libraries to form a runnable jar. The final written report as well as access to the project's documentation will be provided together with the jar to complete the deliverables for the project.

Network Diagram



Gantt Chart

Task Name	Duration	Start	Finish
Parallel Task Scheduling	**28 days**	**20/07/2016**	**26/08/2016**
Project Management	**10 days**	**20/07/2016**	**2/08/2016**
Work Breakdown Structure	4 days	20/07/2016	25/07/2016
Network Diagram	2 days	25/07/2016	26/07/2016
Gantt Chart	2 days	26/07/2016	27/07/2016
UI and Visualisation Design	3 days	28/07/2016	1/08/2016
Software Tools Configuration	3 days	28/07/2016	1/08/2016
Software Architecture Design	4 days	28/07/2016	2/08/2016
Code	**18 days**	**3/08/2016**	**26/08/2016**
Estimation Algorithm	3 days	3/08/2016	5/08/2016
Input/Output Handler	3 days	3/08/2016	5/08/2016
Tree Generator and Traversal Algorithm	4 days	3/08/2016	8/08/2016
Brute Force Implementation	1 day	9/08/2016	9/08/2016
Parallelization Implementation	4 days	9/08/2016	12/08/2016
Visualisation Implementation	6 days	12/08/2016	19/08/2016
Performance Optimization	4 days	23/08/2016	26/08/2016
Code Refactorization	2 days	25/08/2016	26/08/2016
Test	**13 days**	**5/08/2016**	**23/08/2016**
Preliminary Tests	2 days	5/08/2016	8/08/2016
Validation Tests	3 days	19/08/2016	23/08/2016
Performance Tests	3 days	19/08/2016	23/08/2016
Project Deliverables	**6 days**	**19/08/2016**	**26/08/2016**
Final Report	5 days	19/08/2016	25/08/2016
Documentation	2 days	24/08/2016	25/08/2016
Project Compilation	2 days	25/08/2016	26/08/2016