Optimize Prime

Task Scheduling

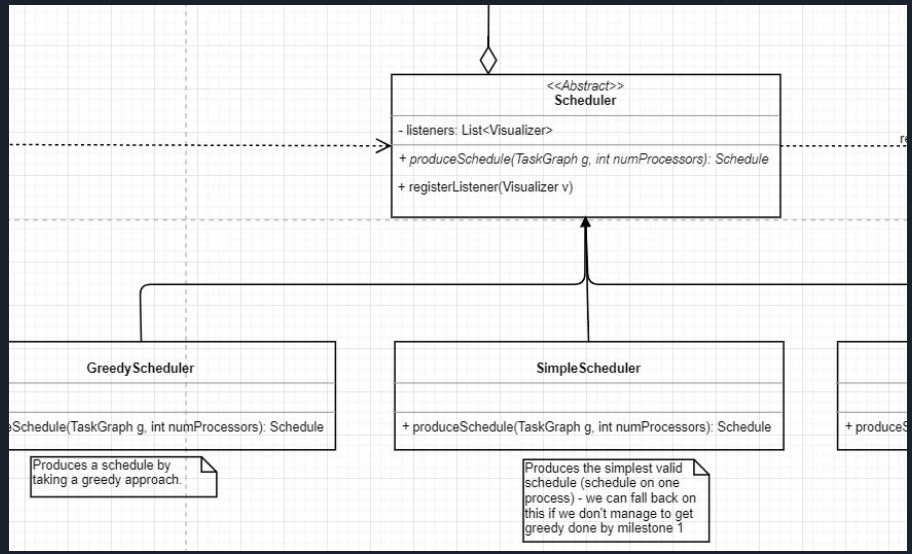
Milestone 1 (almost)

	Task Name
Task 1	**Manage**
Task 1.1	Set up version control, task management, software
Task 1.2	Set up development environment
Task 1.3	Schedule meeting times
Task 1.4	Agree on coding and version control conventions
Task 1.5	Write JavaDoc comments for methods and classes
Task 1.6	Generate JavaDoc files for all code
Task 2	**Research and Design**
Task 2.1	Create a class diagram
Task 2.2	Identify potential algorithm Ideas
Task 2.3	Devise an algorithm which finds an optimal solution
Task 2.4	Identify external libraries for graphs, visualisation, .DOT file I/O and parsing command line arguments
Task 2.5	Identify parallelisation mechanisms available in Java
Task 2.7	Design an appropriate visualisation for the search
Task 3	**Build**
Task 3.1	Create DOT file I/O capability
Task 3.2	Implement algorithm
Task 3.3	Refine algorithm
Task 3.4	Create search visualization module
Task 3.5	Create parallel processing implementation
Task 3.6	Build command line interface
Task 4	**Test**
Task 4.1	Write DOT file I/O unit tests
Task 4.2	Write tests to ensure optimal solution is produced
Task 4.3	Write tests to ensure visualization matches system state
Task 4.4	Fix defects raised by tests
Task 4.5	Write tests to ensure parallel implementation improves upon sequential speed

Plan of Attack for Milestone 1

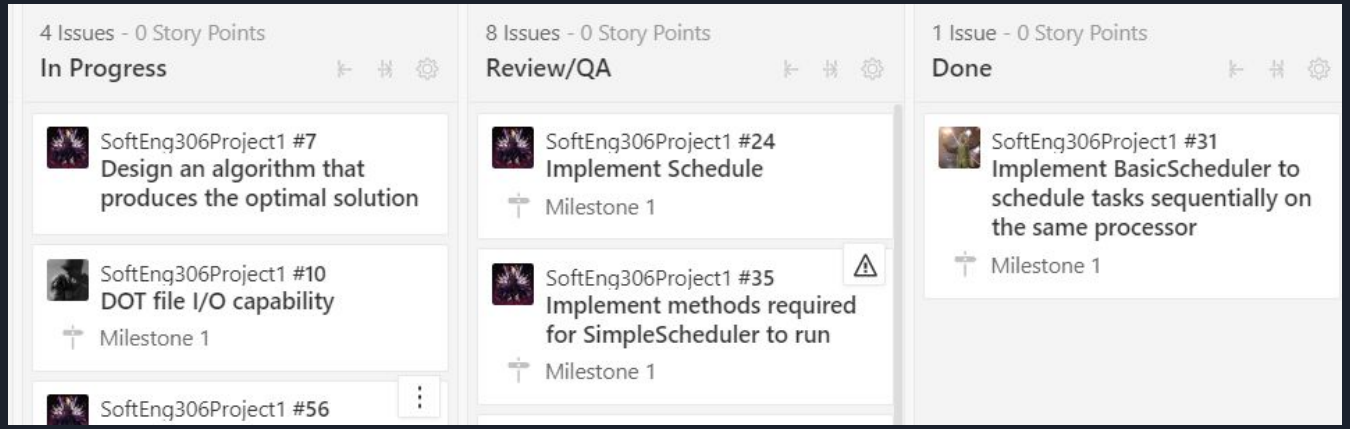
Goal: Deliver a good, but not yet optimal, solution

- Flesh out class diagram
- Implement very basic algorithm as fall back option
- Divide milestone into IO, algorithm and visualisation sections.
- Integrate IO and algorithm modules to produce schedule



Workflow

- Regular meetings twice a week to catch up and delegate tasks on ZenHub
- Each GitHub branch relates to a single ZenHub ticket
- Naming conventions
- Gradle builds the application + run tests
- Travis-CI tests each pull request using Gradle



IO

Command line:

- Parsed command line arguments with CommonsCLI
- CommonsCLI reduced amount of code to write
- Arguments stored in a global module

DOT file:

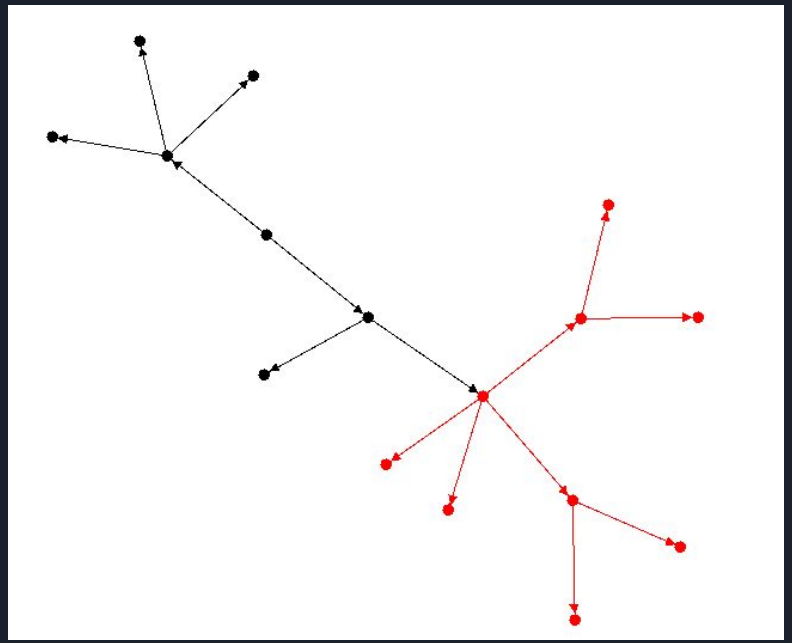
- Read in DOT file
- Graph stored in a model tailored for this application

Algorithm

- Implemented simple algorithm as fall back option
- Scheduled tasks sequentially on same processor
- Implemented a one-pass “greedy” algorithm
- Produced a good non-optimal solution
- Guaranteed functional program by Milestone 1

Visualisation

- Used graph stream library
- CSS integration
- Dynamic updates



Challenges and Successes

Challenges:

- Using git on a larger scale than usual
- Task delegation - some tasks block others

Successes:

- Keeping to the plan
- Design debates => more solid design

Next Steps

Before Milestone 1:

- DOT file output
- Integration of individual modules
- Finalise class diagram

Before Milestone 2:

- Complete visualisation module
- Optimal scheduler
- Refinement and further testing