947G5
Advanced Software Engineering

Group Reflective Essay
2019

Group 2

Contents

Context

This section details the high-level aspects of the project, describing the format of the challenge that the team faced and the implications that certain parameters of the challenge present.

The Task

The task presented to the team is a staggered software development project where deliverables are designed, implemented and tested iteratively. The deliverables each have different timeframes to handle, as well as scaling levels of work depending on the proportion of the individual task with respect to the size of the whole project.

The team consists of seven randomly selected members, each having a different level of experience with the software development process. The team featured a mix of individuals coming from industry backgrounds as well as previous graduates from the University of Sussex.
Naturally a project of this scale involves the use of multiple technologies to fulfil the specification. The task as a whole, developing a cross-platform mobile application, was a novel experience for all members.

The aim of the entire project was to mock a developmental process to create some software, without a specification as to which practice to follow. This gave the team a lot of freedom to craft a freeform methodology to complete the tasks presented by the challenge, choosing a developmental strategy which befits not only the industry practice observed by those more experienced in our team, but also a system that is inclusive of all members' skills and availability.

The team was encouraged to follow common industry practices in the development process such as making use of version control, attempting to work remotely and maintaining a safe and correct development/production deployment strategy. These examples are among many others representing positive strategies to enhance the development process both within a team and without.

The key difference for this challenge is to emulate a learning environment as opposed to a strict production environment, as tackling a challenge of this scale without training members in novel approaches would result in a lower overall quality of the end product. It is not expected for any of the group members to have attained all of the requisite skills prior to the start of the project, and so this is a large factor in deciding how to properly delegate work to individuals of the team.

Organisational Requirements

Effective coordination as a team requires a considerable amount of organisation, derived from the requirements that the project and the subsequent tasks demand. The team originally set out to elect a team leader without an explicit role as project manager, instead deferring the management of the project to the group as a whole. This was later overturned as it quickly proved to be an inefficient method of operation, especially considering the project defines software engineering at scale.

Setting a sense of a learning environment meant that the team had to consider those more experienced as better equipped to undertake a mentoring position. These mentors would act as a main point of reference for team members struggling with issues within the mentor's domain of experience. The idea behind strategising this way is to normalise the levels of experience in the group by supplementing development related issues with proper practice.

It is well established that non-individualistic teams work more effectively by utilising a hierarchy as a 'chain-of-command'. Adopting this as a planning ideal served to prevent overloading a single team member with responsibility whilst ensuring that all team members had adequate access to support.

As development is concurrent across two separate systems, the dividing line between the team categorised individual members as belonging to either the frontend or backend sub-team. As our two mentor-ready team members were experienced in full stack development, communication within these teams was essentially localised until later tasks required inter-group communication to facilitate a working end product.

It is worth noting that the project manager role was essentially unnecessary for the first two tasks due to the workload not requiring the full potential of the team. Full organisational effort came out of redundancy in later tasks which required the design coordination of a system architecture.

Team Organisation

The following section details the day-to-day operation of the team regarding structure and communication. Later sections (inclusive) now detail the implementation of the aforementioned requirements.

Initial Meeting

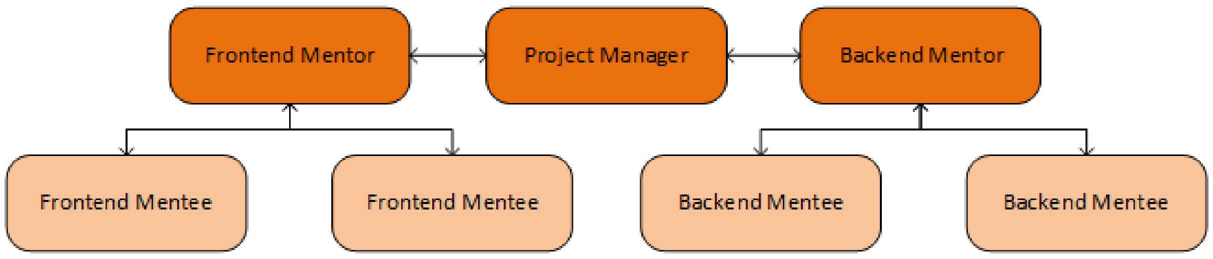
After forming the group, an initial meeting was held to bring into order how the team would operate as well as discussing ideas for the project plan. The first task consisted of not only formulating a project plan, but also agreeing on a repository provider and format. One of the first issues covered was pertinent to the project plan; conflict resolution. Despite a rich discussion for what should be included in a conflict resolution document, there were very few instances of small conflicts and exactly zero substantial conflicts for the entirety of the project lifespan.

On par with discussing ideas for the project plan document, the introductory meeting also gave the team members an opportunity to define the level of their previous experience so that the full capability of the team can be realised and understood. As well as software engineering, team members were encouraged to share all experience that may be objectively pertinent to any potential task presented by the challenge, which may not lay within the bounds of conventional programming skills.

The goal of sharing previous experience was also to provide a scope of technologies that members of the team can be considered proficient in, not limited to just programming languages and software suites but also systematic knowledge relevant to the software engineering process on the whole. This information would later prove useful when determining the high-level design of the project; once revealed.

During the initial meeting, those members coming from a background in industry were asked to put forward their mode of common industry practice to inject some professional standards into our development process from the initial stages of the project. The project manager for the group, being the arbiter of decisional deadlocking, was elected through a democratic vote across all members of the group.

General Team Structure



🟧 Static positions

▨ Flexible positions

The team structure consisted of three fixed positions and four available spaces for members willing to take up roles depending on the requirements of the task. As a task is given to the group, a list of tasks that the team has to complete is drawn up so that individuals can have specific tasks delegated to them. Flexible team members are then encouraged to undertake a task that appeals to their interest, at which point the nature of the task itself (frontend/backend) supplies a mentor that they defer their issues to.

The aim of providing such a free-flowing way to delegate tasks to the remaining members is to provide the opportunity for group members to select a task that they are more complacent in performing. This not only provides an essential base motivation to get the task done, but also supports the production of a learning environment in discovering the specifics in the constituent parts of how a full stack application comes together.

Another consideration to mention for team structure is the communication between the different facets of the concurrent development process. On this level, communication is highly encouraged but unregulated; communication and commitments between a mentor and mentee or two opposite mentees is at their discretion.

Facilitating Communication

The main method of communication that the group used was a WhatsApp group chat. Through this medium, the group was able to organise meetings and communicate issues during day-to-day operations. Whilst it is true that more communication was facilitated through the WhatsApp chat, it is even more prevalent that the group meetings were quintessential to the development process. The main purpose of the group chat is to provide a platform with the benefits of instant messaging.

One aspect of communication that stood out to the group upon reflection is the commitment to organising and attending meetings as a team effort; group meetings were meticulously arranged and attended to ensure that no individual is left ‘out-of-the-loop’.

Not only were meetings held in the timetabled laboratory session every Thursday, but it was also a common occurrence to ask for a group meeting upon the reception of a new task or the completion of a large sub-task. Meetings were rarely missed except for circumstances which were communicated through the group chat, but due to the lack of a tool to contrast group members’ calendars, a lot of pencil scheduling had to be done to finalise a meeting point.

As well as working remotely, both divisions were encouraged to meet on a mentor-mentee basis to work through developmental issues on a person-to-person basis. This was encouraged as it gets progressively harder to describe a thought process over an email or instant message without using code as a collaborative visual aid, especially when two group members are working on the same issue. This touches upon the realms of peer programming but does not quite complete the experience.

System Specifics

This section details the specific tools featured in the architecture of the final design, as well as why they were considered for use in the project. Additional notes are offered as points of interest.

Frontend Development

Node.js

The project makes use of React Native for its cross-platform capabilities between web, iOS and Android. React Native is a JavaScript library as such we used Node.js as a package manager to install various libraries required for the project as well as to maintain project dependencies between machines. This ensured that the project could be developed on any machine within the team as all dependencies where stores in a package txt file which Node.js can detect and automatically reinstall all required libraries onto the new machine.

Expo

We used expo even with its limitation it allowed us to build a cross-platform app completely without touching any native code on any platform. Because we didn't need to write native code through Expo it also meant that we didn't have to have a Mac to develop for iOS and it also made it quicker to go from development to deployment as Expo can run your code in a virtual tunnel for the user to test which features hot reloading on the device too which speeds up development. As opposed to always having to compile binaries with native code. It also made sharing the app around the team easier as it's done via QR codes rather than pre-packaged files. Setting up a project is also a lot easier which speeds up development time and configuring development environments.

React Native

For the project we did not want to develop two separate native applications, so we decided to go for a cross-platform framework. There were a few options out there however we decided to go with React Native as it had the most mature eco-system and two people on the team were already familiar with React so the transition to React Native was a lot easier to get started on development. It also meant that they could share their knowledge with the rest of the team.

Google Maps API

The project makes use of Google Maps API, this was the most logical choice as we where using the Google Cloud Platform for our backend which has seamless integration with Google Maps. Google Maps also has a lot of powerful APIs for geocoding which was necessary for translating street name data into longitude and latitude co-ordinates to be displayed on the map.

Backend Development

Google Cloud Compute Engine

Our backend server is hosted on the Google Cloud Compute Engine, which offers a low maintenance Virtual Machine (VM) running our selected flavour of Linux (Ubuntu 18.04.3 LTS). By using this OS we are reducing the overhead on our machine by only running a CLI, leaving resources free to do more processing.

Our VM has only 1 CPU core and 4GB of RAM. As our infrastructure aims to be lightweight we have decided to use a minimal amount of resources to run our VM. This however is only considerable because we are not working at true scale, but rather emulating large-scale software development.

Python

Our backend scripts are written in Python, which was a common language amongst a considerable share of the group. Python also features a high amount of versatility through both internal and external libraries, which we have used to build a working endpoint for our application.

PIP3 and Packages

The endpoint for our server uses the Flask collection for Python, which provides listeners for standard HTTP requests on an external port. There is also a bespoke Python connector that we use to connect to our MySQL database.

MySQL

The server runs a locally hosted MySQL instance to provide our database. Again, SQL was a commonly understood language across multiple members of the group, and so became an apparent first choice when deciding which database to use.

Interconnection

The client and the server use HTTP POST requests to communicate, using JSON payloads which both endpoints are able to parse efficiently. Asynchronous execution on both ends allows requests to be fulfilled without hanging.

Monitoring System

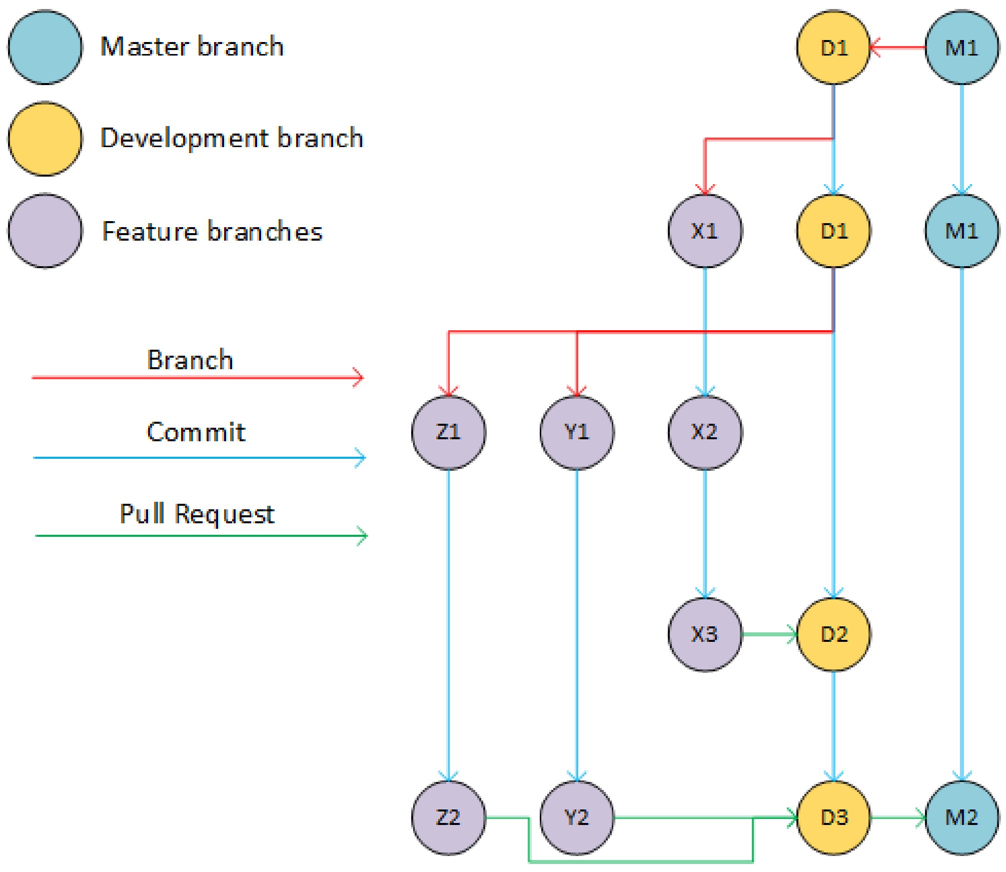
Running concurrently with our server VM, there is also a monitoring VM. This machine runs Zabbix which can monitor various aspects of our system including VM usage, database resource monitoring and application logs. Zabbix is also configured to take a daily snapshot of the server VM in case a rollback is required.

Version Control

To implement a version control system, we made use of a GitHub repo. Following the initial meeting one of the initial tasks was for each group member to familiarise themselves with the Git commands, using a free tool (https://learngitbranching.js.org/) which visualises the tree system that a Git repo represents. The decision to train everybody using this tool was made when it was apparent that more than half of the group had little to no experience with Git or GitHub.

To maintain simplicity in our repository and to facilitate the flexibility of our team structure, we elected to use a monorepo to store our work on. The repository features a master branch used for production/releases only. The develop branch represents an up to date current state of our project, from which features are implemented via branching.

Upon completion of testing on a feature, we merge back into the develop branch using a pull request. By default, the develop and master branches are protected against pushing, and can only be merged into by creating a pull request. This pull request must be approved by two reviewers before merging is unblocked.



Continuous Integration

Our CI consists of a TravisCI script using CodeFactor to analyse our codebase for vulnerabilities, code smells and other common issues such as code styling and hardcoded passwords. The CI script runs on every commit and pull request, generating a report that integrates nicely with the review process to visualise which issues occur as well as when they appear in the code.

Testing

Our backend SQL asynchronous queuing system is tested using unit tests, with an integration test script used to ensure that our endpoint script functions given fabricated POST requests containing both valid and invalid (therefore potentially error causing) payloads.

Frontend testing uses Jest and unit testing as well as system testing by building the app from expo so that we are able to interact just as a user can.

Both endpoints have been meticulously tested by firing fabricated POST requests using utilities such as Postman and Webhook. These were completed prior to connecting new functionality to ensure that both endpoints were communicating in a compatible manner prior to being joined.

Leaning from the Experience

Group Aspects

Communication

On the whole, the group communicated very well. The core of our developmental process was the commitment to meetings observed by all members across the group, such that all group members would endeavour to organise and attend meetings as often as there were issues to bring up in the following meetings.

As well as attending the Thursday laboratory sessions, the group also organised several meetings outside of this time to discuss progress and work allocation to ensure that project work continued to occur on a regular basis.

Division of Labour

Due to the small size of the first tasks in the project, it was apparent that giving members specific roles was highly inefficient to the point that group members found themselves without work to do well before the deadlines. From this we learnt that creating a task pool to assign members to as and when they complete their previous task was far more efficient than simply assigning vague roles to each team member. Development as a result became a smoother process through later stages of the project.

Code Reviews

To facilitate code reviews our group used pull requests on GitHub. Each pull request required at least two approving reviews from members who have not created the pull request. Code reviews were performed by individuals rather than as a group because this demanded too much of a commitment from the group.

Concurrency of Development Stages

The group had issues developing the backend as the system architecture required consecutive development of scripts. This made concurrent group operations difficult to perform as one group member would have to wait on another. Learning this late in the challenge presented the perfect opportunity to reflect on how we could have planned the architecture so that development can be made more concurrent, allowing multiple group members to be fulfilling separate, independent tasks.

Shadowing and Support

Throughout the project the mentoring scheme implemented by our team structure essentially allowed some of our less experienced team members to shadow other members, who were accustomed to working to an industry standard.

One of the strategies employed by the team was to elect support members for each task, as per the conflict resolution section of our project plan. The aim of this was to minimise the disruption in workflow should a member not be able to finish work on time.

Peer Programming

Both frontend and backend mentors were involved in peer programming for a substantial part of the project. Generally, the team experienced a good atmosphere in peer programming due to the presence of different levels of experience aiding in creating a learning environment.

System Specific Issues

Data Loss (Hacking)

During one of the tasks we had discovered a data breach, which resulted in the loss of our price paid data. The reason behind the breach was inadequate security on our SQL database, from which we learnt that stronger security policies were required to properly store and secure our data. From this, we also set up our VM snapshotting so that a rollback could essentially restore the data should we have lost it again.

Handling Concurrency with SQL

While testing our application in the initial stages, we found that too many requests to our SQL connector through our script would cause the connector to drop the connection to the database. This was very problematic in that further requests to the already running script would cause errors in our backend until the connection was re-established by restarting the script.

In the spirit of allowing concurrent requests to be fulfilled, the team came up with the idea to implement an asynchronous queueing system after a design meeting as opposed to allowing and detecting a script failure and restarting accordingly.

Heatmaps with React Native

Whilst prototyping a heatmap using react native, our frontend developers found extensive discussion regarding the non-functionality of the Google Maps Heatmap Layer when running through React Native. As a result, the group designed a different system to imitate heatmap functionality to fulfil the specification.

The scope of this issue is that team had to adapt to a compatibility issue with our implementation against the specification of the project, as quickly as possible. On reflection it was suggested that more research into Google Maps' functionality with React Native would have uncovered this issue far earlier than the prototyping stage.