Software Development Plan

Dota Track

Michael Eaton, Hazen Johnson, Sarah Witty

September 30, 2014

Contents

1 Summary

Dota Track is a Dota statistics application distinguished it's ability to give historical data for a player's history. This will allow players to track how they have improved over time, and as a result, to become better players. The main client for this Software Engineering 2 project is the Letu Moba Club; although the application will be general enough for any Dota player to use. This document lays out the development plan of the **Dota Track** application.

1.1 Development Plan Goals

This software development plan aims to:

- Organize the team's collaborative and individual time to most efficiently complete tasks required for development.
- Provide milestones against which development progress may be compared.
- Ensure that development proceeds according to plan by verifying products against milestones.
- Provide expectations for communication between team members and the client to promote understanding at all stages of development.
- Develop a quality assurance plan to maintain a certain level of quality in the final product.

1.2 General Goals

The general goals for **Dota Track** are laid out in (SRS 1.2).

1.3 Constraints

- Development must be complete by October 31.
- No budget has been allocated for this project.
- Dedicated time will be limited to x hours per week per team member.
- Deployment will be limited to available servers, such as cs-lab.letu.edu.

1.4 References

This document is based on the **Dota Track** SRS and SDD. References to these document will be noted as (SRS X.X.X) or (SDD X.X.X) indicating relevant sections in the SRS or SDD respectively.

2 Project Estimates

Assuming that each difficulty point will take two hours to complete:

	Controller Application	Database	Ajax Module	Graphing Module	Interactivity/ Responsiveness	Stage Total
1a. API Pulling	3	n/a	2	n/a	2	7
1b. Database	4	2	2	n/a	n/a	8
2a. Display Match (Matches Page)	3	n/a	2	n/a	4	9
2b. Display Player Stats (Statistics page)	3	n/a	2	4	4	13
3. Parse stats by Hero	2	n/a	1	1	3	7
4. Tag Matches, Parse by Tag	2	1	2	n/a	3	8
Module Total	17	3	11	5	16	52

Figure 1: Difficulty Estimation Table

Minimum:

1. API Pulling and Database Setup = $(7 + 8) * 2 = 30$ **hours**
2. Display Matches and Player Stats = $(9 + 13) * 2 = 44$ **hours**
3. Parsing Stats by Hero = $(7) * 2 = 14$ **hours**
4. Tag Matches, Parse by Tag = $(8) * 2 = 16$ **hours**

Maximum:

1. API Pulling and Database Setup = $(7 + 8) * 4 = 60$ **hours**
2. Display Matches and Player Stats = $(9 + 13) * 4 = 88$ **hours**
3. Parsing Stats by Hero = $(7) * 4 = 28$ **hours**
4. Tag Matches, Parse by Tag = $(8) * 4 = 32$ **hours**

Total:

$$\sum \text{Minimum} = 104 \textbf{ hours}$$

$$\sum \text{Maximum} = 208 \textbf{ hours}$$

Figure 2: Converting Difficulty Estimate to Hours

These estimates lead to the conclusions that:

- Each team member will be expected to work at 10-20 hours a week.
- The project can be completed before October 31, with some minimal amount of breathing room.
- A large part of the complexity of the project will be focused on early-mid development.

3 Risk

3.1 Risk Description

- Product risk - **Dota Track** will not provide any more functionality than is already available through tools such as DotaBuff.
- Customer risk - the customer will be unable to meet with the development team on a regular basis due to other projects.
- Development risk - the chosen frameworks and environment will not easily provide the functionality needed for **Dota Track** to operate effectively.
- Employee risk - the development team may be unable to learn and use the chosen frameworks to implement **Dota Track** , resulting in sub-par implementations and behind schedule completions of software modules.

- Process risk - the results of the **Dota Track** application will be inaccurate, rendering the final product unusable.
- Product size risk - **Dota Track** will attempt to support too many features and require a massive back-end far beyond the time capabilities of the development team before the given deadline.
- Technology risk - the APIs used will change, rendering the **Dota Track** application unusable.

3.2 Risk Management and Quality Assurance

Methods of risk management include:

- Requirement/design driven testing.
- Weekly meetings and code review.
- Frequent (weekly at minimum) client meetings, validating software products.
- Working build server (cs-lab.letu.edu).
- The master branch of the source tree should always be usable.
- All code syncing should be done through the master branch.
- Kanbann and Burndown charts.
- Mid-development re-estimation - the estimates given in the Estimates section (2) will be re-evaluated half way through development, during a weekly meeting.

4 Project Schedule

Schedule is laid out in accompanying Microsoft Project file. Please examine this file to find the exact scheduling information.

5 Project Organization

5.1 Team Organization

- The development team will be a headless group of developers which organically makes decisions together.
- Each individual developer is empowered to make critical decision by themselves; however, in such cases, the other developers should be informed as soon as possible.
- Individual team members will be assigned specific modules in each segment of development by the team.
- Team members will be responsible for testing their own code before the weekly review.
- Weekly code reviews, by the entire team, will take place to ensure code quality.

5.2 Resource Organization

The following resources will be used to aid project implementation:

- Development environments and technologies
  - Kohana - PHP framework
  - MySQL - Database engine
  - D3 js - Javascript graphing library
  - jQuery - Javascript utility library (for Ajax and DOM manipulation)
  - Apache - web server using PHP (modphp)
  - Mobile devices (Windows Phone, iPhone) - possibly an Android device if discovered.
- Deployment environments
  - cs-lab.letu.edu - Used as a test deployment server. The test deployment should update semi-automatically to the master branch of the source repository.
- Code repository
  - GitHub - used for source hosting and version control.