Multimedia Services : Product Planning

Group BeNine
TI2316 Context Project

Bryan van Wijk (bryanvanwijk, 4363329)
Dorian de Koning (tcmdekoning, 4348737)
Ege de Bruin (kedebruin, 4400240)
Jochem Lugtenburg (jlugtenburg, 4370805)
Naomi de Ridder (nderidder, 4383109)

May 4, 2016

Supervisor: Dr. Cynthia Liem
Software Aspect TA: Valentine Mairet
Context Aspect TA: Alessio Bazzica

Delft University of Technology
Faculty of EEMCS

Contents

1 Introduction

This document contains information about the planning, and constraints involved with the context project. The contents of the documents are divided in five main chapters. First, a high-level overview of the product including requirements will be given. Second, user stories are given. Third the initial release plan is given. Fourth the roadmap, which is the global overview of the planning for each sprint is given. The final chapter contains the Definition of Done.

Context analysis

Currently, a script is written in a simple database or spreadsheet, instructing the camera operators on how to program their Presets. A possible scenario involves a recording session of a concert at the Concertgebouw. Before the concert, rehearsals take place. The client also takes part in these rehearsals by testing the script created by the director. Before rehearsing, the camera operator programs the required presets into the camera controller. This is a very intensive task, requiring a lot of buttons to be pressed. During the rehearsal and final performance, these presets have to be called, requiring a similar procedure. After the rehearsal, the camera operator knows which presets to call. Since the scene changes because people move, he needs to adjust the presets while the concert is in progress before starting the actual recording. Due to the intensive task of calling a preset, the camera operator has only little time to adjust the presets.

2 Product

This section starts with a short high-level description of the product that is envisioned during this project. After this the requirements of the product are described using the MoSCoW Method.

2.1 High-level product

The product the team will develop is mainly used to support the camera operator. The system will contain a web interface in which it is possible for the camera operator to control the different camera's. Besides controlling the camera it must be possible to set and load presets. Presets are stored viewpoints which contains the position of the camera at the time the preset was set. Additional to this simple presets, presets could also be more intelligent by tagging a preset with keywords representing what the preset scene contains or describe camera movement between two set positions.

2.2 Requirements

The requirements are described using the MoSCoW Method method. The MoSCoW method is a widely used standard in the industry and involves prioritized requirements. The requirements are categorized in four categories, 'Must', 'Should', 'Could' and 'Would/Won't'. If a 'Must' is not included in the final deliverable, the project is not considered a success.

2.2.1 Must have

- A user shall be able to remotely control the camera's.
- A user shall be able to select a camera to control using a single tap on a user interface.
- A user shall be able to use the application using a touch sensitive display.
- A user shall be able to load a predefined preset.
- A user shall be able to store multiple Presets per camera.
- A user shall be able to store multiple presets in a database so they are persistent over different program sessions.
- A user shall be able to tag a viewpoint and store it as a preset.

2.2.2 Should have

- A user shall be able to see the live feed of more than one camera in one screen.
- A user shall be able to add new Presets to the system during a performance.
- A user shall be able to share Presets between camera operators.
- A user shall be able to get an overview of the system connection status in the user interface.

2.2.3 Could have

- A user shall be able to tag a preset with information regarding its contents.
- The system shall be able to provide suggestions about similar possible presets in the viewpoint of a camera.

2.2.4 Won't have

- A user shall be able to create a digital script of camera views which can be executed automatically.

2.3 Motivations of priorities

Must haves

Among must-haves, the team considers essential elements of the final product. This means if they are not included in the final product, the product is not useful or not an improvement to the current situation. These must haves include camera operations, control and intelligent presets.

Should haves

Non-essential features improving user experience are marked as should-haves. It would be nice to have them but the product is not completely useless without them. These include editing camera presets during live performance.

Could haves

Features expanding the capabilities outside of the main scope of the product are considered could-haves. In the event of extra time, the team may consider to implement some of them.

Would/Won't haves

In the case of a feature which will definitely not be implemented, we mark it as a won't-have. This includes a script editor, since the client is already in possession of such a program.

3 User stories

In this section user stories are described, a user story is a short description of what a user wants. The form in which they are written is "As a <role>, I want <desire> so that <benefit>"

User story 1

As a camera operator, I would like to be able to control camera's using a web interface on my tablet computer. On this web interface I would like to select a camera to control with a single touch, so that I can spend more time adjusting the preset before starting the camera recording.

User story 2

As a camera operator, I want to store multiple presets per camera. Also, I want to be able to change the saved presets and load one of the saved presets, moving the camera to the desired position so that I don't have to press multiple buttons.

User story 3

As a camera operator, I want to be able to create a preset by taking a picture of the viewpoint as a tag providing a visual clue of the preset so that I can recall the preset easily.

4 Initial release plan

Every week there will be a release after a sprint ends. This release includes the new features finished in that week. The final release is planned on June 17th. For building the system continuous integration is used. This means that every version of the system is automatically built and you can see directly if there is a build failure. The team aims for the final product to be easy to deploy, requiring as less installation steps as possible.

5 Roadmap

Now, a general overview of each sprint will be given, formatted in a comprehensive list.

Initial Phase

The initial phase involves a visiting the client and designing the high-level architecture of the program.

Sprint 1

- Simple web interface and basic server infrastructure.
- Create product plan and vision.

Sprint 2

- Implement basic functionality for Presets.
- Implement basic control of the camera's
- Create intelligent preset plan.

Sprint 3

- Finish implementation of basic functions.
- Begin implementing intelligent presets.

Sprint 4

- Implement intelligent presets.
- Preset creation interface.

Sprint 5

- Change product based on user-feedback.
- Expand preset creation interface.

Sprint 6

- Improve synchronisation between multiple interfaces.
- Improvements according to user-feedback.

Sprint 7

- Extension, used for user-feedback or additional features encountered in the process.
- Acceptance testing.
- Improvements based on acceptance tests.

Sprint 8

- Finalize product, resolve bugs.
- Prepare product for release.
- Complete final report

6 Definition of Done

To conclude this document, a set of rules will be given which are used to judge if a certain task from the sprint plan can be marked as 'Done'. This set also applies to integration of branches into the main branch, and the review of deliverable documents.

6.1 Sprint Plan Tasks

A sprint plan task is considered 'Done' if it complies with the upcoming three rules. First, the work done should match with the description mentioned in the Sprint Plan document. Second, if a user story is involved, this user story must be fulfilled and properly tested. Finally, the majority of the team members should agree upon marking the task as 'Done'.

6.2 Features

To mark a feature as 'Done', a predefined procedure has to be followed. All features implemented, must be implemented in a branch which is not 'master'. The feature(s) implemented on this branch should be thoroughly tested, with a line coverage of at least 75% and must be following the code conventions dictated by the Checkstyle plugin in Maven. Errors of warnings generated by PMD and FindBugs should be eliminated if possible. However, we did not follow a strict set of rules but choose what conventions we wanted to apply. We did this because some of the rules are not important for this project. Proper documentation should be added to ensure that people who are not involved in the project, or people who haven't been working on a feature for a long time can understand the decisions and methods used to write the code, so they may extend it. During integration, in the form of a GitHub pull request, the team reflects the rules above. If the majority of the team agrees, the feature branch may be integrated into the master branch, marking the feature as 'Done'. To build and test the system a continuous integration tool called Travis CI will be used.

6.3 Deliverable Documents

Deliverable Documents must comply with the points described in the 'Deliverables' section of the general context project guidelines document (the document should for example contain a title page, our names, netid's and a table of contents). Each document should also be written in clear, formal English language.

6.4 Sprint and Final Product

A Sprint is successful if all tasks are marked 'Done' or if tasks are left undone for a good reason. If any problems occur, the team must reflect on them during retrospective and improve in the next iteration. A successful Final Project should be accepted by PolyCast, the client and must meet the quality aspects described in the previous subsections. The must-haves described in the Product Backlog are very important. If the team fails to include one of these cases the product might not function, or it may be unpleasant to use. To improve the user-experience, a reasonable segment of should-haves should also be implemented.

Glossary

branches a branch is a copy of the system so modifications on the system can happen in parallel. 10

Checkstyle a development tool that helps write Java code that adheres to a coding standard. 10

FindBugs a static analysis tool to look for bugs in Java code. 10

MoSCoW Method is a widely used standard in the industry and involves prioritized requirements.. 4

PMD a code analysis tool for your product's source code. 10

Presets a stored viewpoint of the camera at the time the preset was set. 3, 4, 8

tablet computer is a portable handheld computer device, usually with a touchscreen. 6