Product Planning, TI2806

Blazin and the Goons

Authors:

Shane Koppers,	4359062
Floris List,	4380258
Hidde Lycklama,	4397614
Joris Oudejans,	4393694
Jordy Weijgertse,	4247124

Contents

1 Introduction
2 Product
    2.1 High-level product backlog
    2.2 Roadmap
3 Product backlog
    3.1 User stories of features
    3.2 User stories of defects
    3.3 User stories of technical improvements
    3.4 User stories of know-how acquisition
    3.5 Initial release plan
4 Definition of Done
    4.1 Features
    4.2 Sprints
    4.3 Releases
5 Glossary

1 Introduction

The following document describes the product and its planning that will be developed by group ‘Blazin and the Goons’ within the scope of their Contextproject. In this project, the end user is PolyCast, a company that specializes in live video and audio recording, specifically focussing on classical music.
During the introduction at PolyCast on April 19th, it quickly became apparent that the recording process was a labor-intensive action for PolyCast, even though it is something they do regularly. Therefore, it was clear that their work could me made more efficient by creating a product that improves the recording by reducing the need for manual labour.
In this product planning we first cover the product and its features. Next, we go over the product backlog and, finally, we go over our definition of done for the various features.

2 Product

This chapter describes the details of the end-product by covering the high-level product backlog. After this, the roadmap, which specifies which features will be implemented in which sprints, is laid out.
The product can be seen as a combination of features. The features that make up the product are ordered and ranked by importance, via the MoSCoW-method. This is done because some requirements are more important than others. Therefore, their corresponding features are too. The MoSCoW-method provides a logical ordering for importance of features:

Must haves features that are essential for the application to function in a meaningful way.
Should haves features that have a high priority and should be done, but could be satisfied in a different way if needed.
Could haves low priority features that could be implemented if time permits it.
Won't haves features that will not be implemented, but could be features for a follow-up project.

2.1 High-level product backlog

The product will be a tool to manage a script before and during a recording. Because the system will understand the script, it can help during a recording by executing certain tasks, such as moving the cameras to presets. Furthermore, it optimizes the communication and real-time transitioning in the script by making sure everybody's interface is on the same page.

Must haves:

- User-interface that can be used by all members of the PolyCast team
- Tool to create, manage and remove scripts
- Script database with structures for script actions, cameras and presets for cameras
- Real-time active camera view
- Web-based interface optimized for tablet screens for recording mode that is synchronized with the other operators and director
- Director controls to control the script during recording
- Script actions automatically controlling cameras via the camera's API

Should haves:

- Camera position presets
- Camera preset creation tool
- Scheduling system to match script actions together with camera presets in order to determine a working order and find conflicts in a script.

Could haves:

- Exporting the recorded sequence in a file format that defines only the sequence of video (and not the video itself). This should be a format, like xml, usable by PolyCasts' video editing software.

Won't haves:

- Elaborate learning algorithm to grade certain camera positions according to given heuristics to automatically decide an optimal camera position.
- Algorithm to synchronize camera settings like warmth or brightness.

Both features won't be implemented because they would take a lot of development time while being too far out of the product's focus.

2.2 Roadmap

The planning is divided in weekly iterations, called sprints. In each sprint we focus on different things. In the first sprints the focus will be on getting the application's main features up and running, while the later sprints will be focused toward less important features and details.
At the end of each weekly sprint the project should be in a state that all features for that week are done. At this time, friday evening, the next major release has been finished.
The goals for a sprint release are to finish all tasks in the backlog completely, including the testing of the features. This way, a sprint can be really finished and features are really done.

The following table describes focus points for each group of sprints

Sprint(s)	Focus
1,2	Set up project, developer tools, global framework, decide on project approach
3,4	Work on important, main features, user-interface, communication
5,6	Refine features, work on detail, make changes based on feedback of client, add new ideas that come up
7,8	Work on quality, finalize system, make product stable for release, make changes, based on feedback of client
release	A working product with as many complete features as possible

3 Product backlog

3.1 User stories of features

1. As a user, I want to add, edit and remove my scripts and script actions in a web-based interface, so that I can manage my scripts.
2. As a user, I want to run scripts with multiple people so that I can communicate the current action in the script and that everyone is one the same page.
3. As a user, I want to be able to control the script as a director, so that I can control the recording flow.
4. As a user, I want to be able to automatically run a script, where cameras are put in the right position automatically, so that I can focus on the details of recording.
5. As a user, I want to be able to change presets and scripts while recording, so that I can be flexible in recording.
6. As a user, I want to have my screen updated to the latest changes in the recording, so that everyone is in sync during a recording.

3.2 User stories of defects

Defects are also defined as user stories in sprints in order to fix them. They will be of the same format as the regular user stories. An example is:

- As a user, I want to update my main script view when I add an action so that I see the updated script.

3.3 User stories of technical improvements

Technical improvements will, like defects, also be defined as user stories.

3.4 User stories of know-how acquisition

Learning new things is often embedded in the execution of certain tasks for user stories, so they will rarely have their own user story. However, it might occur in the earlier sprints to gain knowledge about the Play framework and camera API's.

3.5 Initial release plan

The initial plan is to release the product with all features done that are in the must, should and could have categories in the high-level product backlog at [2.1].
The minimum releasable features can be a small subset of the above features. However, at least some functionality must exist to manage scripts, as this is the basic functionality of the product. From this basic functionality, other features can easily be added on.

4 Definition of Done

This chapter describes what needs to be done on the different subjects in order to consider it as done.

4.1 Features

A feature is considered to be done when the following conditions are met. It should be explained when some condition isn't fully met.

- The feature is fully developed along with all it's sub-functionality in order to realize the feature.
- The code connected to the feature must have the following properties.
    - Enough tests have been developed to test this feature (both unit and integration tests) such that the code coverage is 75%. An exception can be made for user-interface related functionality.
    - Checkstyle does not give any warnings for the code. Checkstyle will use a custom ruleset so that we won't lose too much time on some unnecessary rules.

    - The code must accord to high software architecture standards

This ensures that the feature is ready to be added to the main project. The feature works, is well documented and is (at least) reviewed by two other members of the team.

4.2 Sprints

A sprint is done when the week is over. All tasks in the sprint backlog that are not done (if any) are moved to the next sprint if they are still relevant. Every sprint should release a product which is an improvement of the product of last week. The release of the improved product should have a version number and a tag before it can be considered done.

4.3 Releases

A release is done when the code is stable and the necessary features are implemented. Moreover, the stakeholders should be satisfied with the product. Finally, the code should also pass the static tests and thresholds.

5 Glossary

AngularJS - Front-end javascript framework
Bug - A flaw in the code that causes errors.
CheckStyle - A static code analysis tool that lints the code for writing style.
Java - A commonly-used object-oriented programming language used in our project
Play framework - A webserver framework based on Java / Scala
PolyCast - Company that is the end-user of the application
Scala - A programming language that compiles to JVM bytecode
Sprint - A development cycle in which tasks in the backlog are performed
TypeScript - A programming language that transpiles to javascript ES5/6.
UI - User Interface. This is the interface users see and use to operate the system.