Final Report - Draft

The First Order

Chris Berg 4216776
Martin Koster 4371011
Hung Nguyen 4232410
Christian aan de Wiel 4396286
Ruben Wiersma 4214250

17 June 2016

TU Delft

0 Abstract

This is the final report for the 2016 context project in the Multimedia Services context. Team *The First Order* has developed a web application for PolyCast that aids their team in creating scripted live recordings and streams of concerts. The web application's unique selling point is its interface for creating and visualizing a script before the production is made. This is done through a time line that is mapped to a score and a map that visualizes the concert hall. Other features include editing maps, managing projects, exporting projects and a live mode that can be used during production. This report gives an overview of the developed product, detailed information on all its functions, reflections on software engineering and interaction design and an evaluation and outlook for the future.

Contents

1 Introduction

In the spring of 2016, classical recording company PolyCast came to us with the challenge to develop an application for them that could improve their workflow. PolyCast is specialized in making live recordings of concerts all around Europe (and the world). When they prepare for one of these shows, they usually start with the score of the music. From this score, they derive a script that is printed on paper and used in live production. Afterwards, the script can be used to edit a final video of the concert. Currently, scripts are created in FileMaker and then distributed on paper (and possibly PDF). This is where The First Order comes in.

Our task was to create an application that could save a script in a database that could be accessed by every user in the workflow. A director would create a script, a score reader could progress through the script during production, a camera operator would see all cues relevant to him and all other users would be able to export the script to PDF, so the product could fit in the current workflow well. This all would have to be done in a way that is user friendly and intuitive; one should not need a lot of time to understand the software and using the product should be a joy instead of a chore.

The First Order is proud to present *The First Script* as the application that fills the gap and meets these requirements. We will give a general overview of the implemented product in chapter two, followed by a reflection on the software engineering side of this project in chapter three. In chapter four, all the components are described in a more detailed way and in chapter five we take a look at the interaction design aspect of our product. This is followed by an evaluation in chapter six and an outlook to the future in chapter seven.

2 Product Overview

This section provides an overview of the developed and implemented software product. In chapter four this is done in more detail to explain the specific functionalities of the product. This section also explains the ways in which the product will be used.

The First Script was made to support PolyCast in their entire workflow concerning scripts. The product's unique selling point is the intuitive interface for creating and visualising a script the ability to save the script to a database and how it lets the user view, export and download the script in multiple ways.

In the scripting interface, a timeline is 'mapped' to the score of a concert. Currently, this mapping is done in a straighforward, simple, way: a bar on the timeline represents a bar on the score. The user can also view a map of all cameras and player sections present in the concert. By double-clicking on a point in the score (timeline), a user can add a cue that holds an action and the player section that is in the shot. This cue can later be changed by dragging it around on the timeline. When the user scrolls over the timeline with a timebar, the map shows which cameras and players are 'in use', giving a visualisation of the script. When the script is finished, the script can be exported to a PDF or XML file that can be used during or after production.

The product has an extra view for live production. In this view there is a list of all the cues for a selected camera in one overview. A score reader can progress the current location in the script, which is updated for each camera operator. The current cue is highlighted so it is clear which cue is active. Any mistakes can be undone with the "previous" button.

To make it possible for PolyCast to work on multiple projects at once, we have implemented a project-based workflow: users can create and load projects where maps and scripts are saved. These are all loaded into the application when a user starts it up and selects a project.

3 Software Engineering

Developing this product started with deciding what language and frameworks would be used to aid development. It was decided quite quickly that a web app would be the best fit for PolyCast, as a web app would be easy to use on a desktop or laptop computer, but also on a mobile platform like a tablet. A web app would also automatically work on all platforms with a web browser without having to refactor anything. Therefore, we set out to find the right tools to make this work. Java was chosen as the backend, as this was the language the team members were most comfortable with and because SIG would be able to validate the code well. As we were looking for the right frontend framework (sprint 2), we stumbled upon JHipster, a framework that puts together Springboot (a Java backend framework) and AngularJS (a JavaScript frontend framework). Springboot had already been chosen and partly implemented because of its extensive documentation and good reviews and AngularJS was also known to have a lot of online support. Using JHipster made sense, because it meant we could dive right into developing a working application and could learn how the frameworks worked on the go.

As we worked with Springboot and Angular and the classes that JHipster provided, we found that we had to learn a lot about how to develop inside this environment. It took some time to get used to working with callbacks, RESTful APIs and GET and PUT requests, but eventually all team members got the hang of it. The biggest problem was in getting the frontend to interact well with the backend and load and save the correct data. This resulted in some sloppy code in the frontend that was later fixed in the sprint 5 and 6.

We wanted to have clear software architecture with strict guidelines and high software quality. Therefore, we sought to adhere to the guidelines provided by the John-Papa Angular styleguide, like having short JavaScript functions (Carraway, 2016). This guide gave us directions in how to code cleanly, but also the why. Because we tried to adhere to these principles and the functionalities JHipster gave us so tightly, we had to rework some code a couple of times, but it also resulted in a clean architecture that has a clear divison of responsibilities and functionalities.

We have put extra effort in testing our software to a high extent. As we wrap up product development, 85% of the software has been tested and validated through a mix of mostly unit tests and some E2E tests.

Concluding, we can see a steep learning curve in the beginning of the process and a gradual, but consistent growth in functionalities, loosely following the roadmap we set up, and quality, because of the foundation was setup in the start. This all was validated with a testing suite that gave us a rational ground to trust our product.

4 Functionalities

In this section, the functionalities of the product are described in further detail. The functionalities are grouped together to get a clearer overview of them. The first category will describe all functionalities of the script, the second category of the map editor, the third category of the project management and the last one of the live view.

4.1 Script

A script consists of a timeline that is mapped to a score. This mapping is still quite simple (bars all have the same size) as the image recognition required for automatically mapping a timeline to a score would fall out of the scope of this project. This timeline holds cues that are set on bars. There is a timeline for each camera in production, so a director can see at a glance what cameras are used at what moment. Double clicking on a timeline creates a cue. This cue can have a camera action and a player. When a player is selected in the map and a new cue is created, this player is automatically filled in for that cue.

After adding a cue it is possible to move a cue to a different camera or time point just by dragging it to the wanted position. The duration can be adjusted by dragging the sides of a cue, the information of an existing cue can be updated by double clicking that cue and finally a cue can be deleted by pressing the delete button when a cue is selected.

During scripting, it is possible to drag a bar across the timeline to see what will happen at a given moment. When, for example a cue is setup to point camera 1 at the conductor, camera 1 and the conductor will be highlighted in the map.

When the script is finished, the user can export the script. This can be done in two formats: PDF and XML. These exports contain all information about the script and relating cues.

4.2 Map editor

In order to customize the map that is shown in the scripting page, we have built the map editor. The map contains the positions of the cameras and players of a certain concert. First, every concert hall is different. Therefore, it is possible to change the concert hall background. This photo can be taken during a rehearsal

and uploaded to the map editor. If no map is uploaded, the default concert hall is the *Concertgebouw*.

In this map view, new cameras and players can be added. To add a new camera or player, the user can double click in the map and choose to add a camera or a player. The position in the map is recorded with x and y coordinates and the camera or player can be named. The camera can also store a camera type and lens type, because different cameras can be used for different purposes. After adding an entity, the information can accessed by selecting the specific camera and the information will show up. All the cameras and players are put in an overview next to the map. In this overview a specific camera or player can be selected to see the information about it.

The user can change the position of a camera or player by dragging it to the correct position on the map. Finally, it is also possible to delete an entity. This is done by pressing the delete button next to the camera or player in the overview.

4.3 Project management

Each scripts and map is different for every concert. Therefore, the user can store maps and concerts in projects. When the application is started and the users has logged in, a user can create a new project or open an existing one. All new and updated information in a project is saved automatically, so when an existing project is opened, all data is preserved. When there are no maps or scripts, the user can directly create and load new maps and scripts and a user can also use a map or script from another project in a new one.

4.4 Live view

The final aspect of the application is the live view. This view is created for the score reader and camera operators who control and use the cues during live production. In this view, all the cues for the active project are shown in order and a camera operator can select which cameras they are operating so they will only see those cues. The next button can be pressed by the score reader to indicate that the current cue is terminated and the next cue is active. When the script is progressed on one device, all other views are updated simultaneously, so that every team member's view is synchronised with the others'. If a mistake is made, the score reader can use the previous button to go back to the previous cue.

5 Interaction Design

The software was designed from an interaction design perspective. One of its unique selling points is that its way of creating a script and visualizing its actions is intuitive and helps the user to visualize a concert beforehand. During development this was approached by emulating behaviour users know from other software, testing the software and by thinking through scenarios. It is now beneficial to gather feedback from actual users to see if the interaction design efforts paid off and to give pointers for further development of the software.

5.1 Method

Since the results from this experiment would be used to improve the software and because the software was still in development at the time this experiment was taken, we have done a formative experiment with the research question *can users independently use our software and how do they experience the product?* The focus was on four parts of the software through one overarching task *to create and export a script.* The underlying parts are: loading a project, the map interface, the script interface, and exporting a project.

The user test was structured according to Nielsen's studies (Nielsen, 1993). First the test was *prepared* by filling the software with demo data, and describing the tasks for the candidates to complete. The tasks can be found in appendix A. We went to the IDE faculty, because the students there were used to doing user experience tests and were similar to our target users with regards to software experience and technical skills. When a candidate was ready, they were *briefed* about our test, indicating that they were still using software in an early stage of development, and telling them to relax, take their time and think aloud. Then the actual *testing* was done by letting them use the software. Notes were made about the time each action took, wether an action was completed (based on ISO 9241-11 standards for effectiveness and efficiency), and the thoughts the candidates expressed by thinking aloud. These note sheets are included in appendix B, the note taking form can be found here: `http://goo.gl/forms/IBT79pV78deP4Yxf2`. Finally, the candidate was *debriefed* by asking them to describe their experience with the software with a Manikin (Bradley & Lang, 1994), and asking them about any suggestions they might have. This test was first piloted with one candidate to iron out any kinks in the testing procedure and then really tested with 11 students from the IDE faculty at the TU Delft.

Since each task was clearly defined, there was no need for a separate definition of

when a task was completed. We would intervene if a candidate needed more than 30 seconds and was asking for help. If the candidate asked for help before the 30 seconds, they would be motivated to try a little longer.

5.2 Results

Below we have visualised the percentages of the actions that were done without help (figure 1). Most actions could be completed without any help, so we have not included these rates. During testing, we found out that timing a task did not work as well as we had hoped. Some users spent a lot of time explaining their thoughts, therefore making the time spent on an action meaningless. For this reason, we have not included the results of this part of the test in the report. All the results of the test can be found in appendix B. Finally, we have visualized the user experiences on a linear scale (figure 2-4).

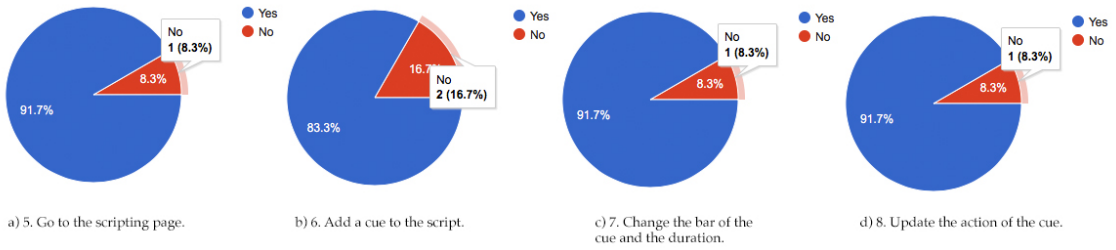


Figure 1: The rates of success for tasks.

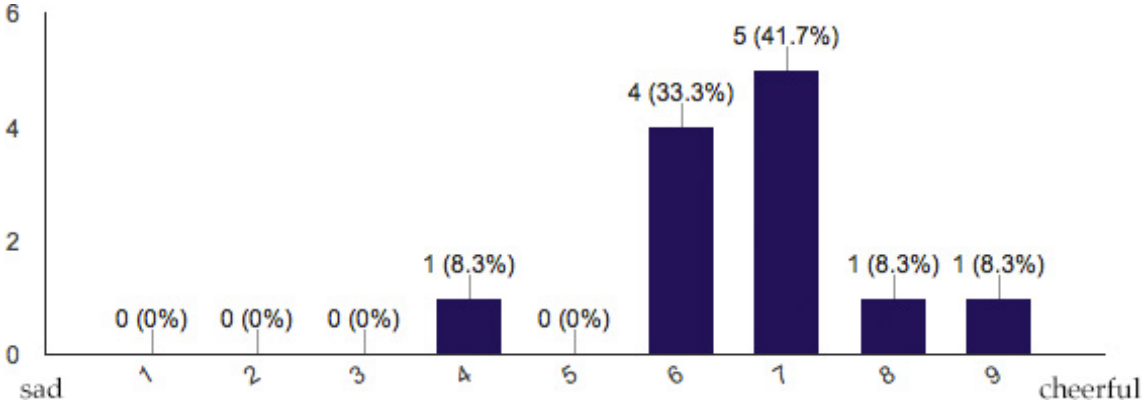


Figure 2: The mood users experienced with our software.

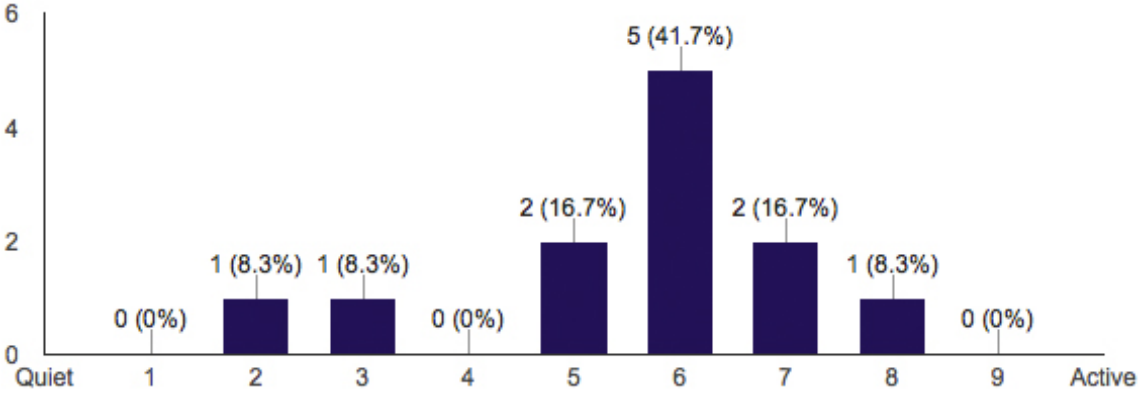


Figure 3: The activity users experienced with our software.

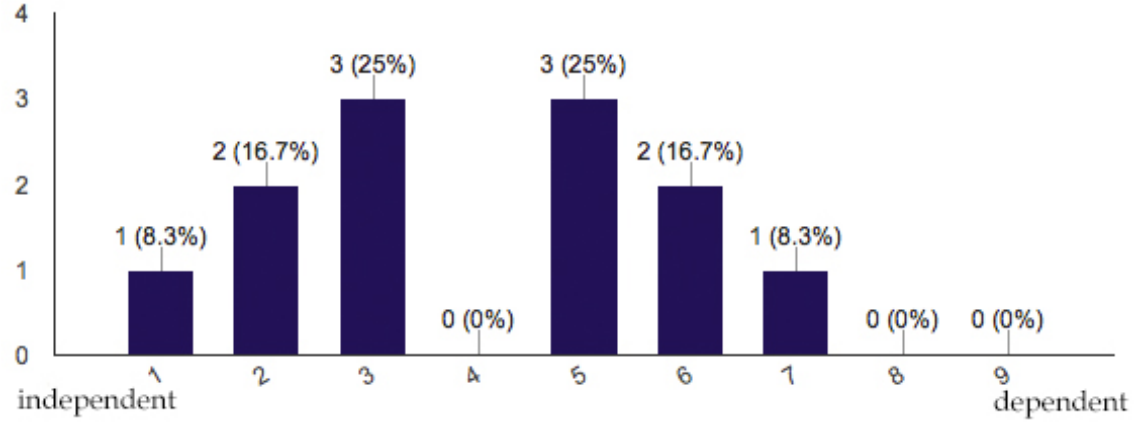


Figure 4: The dependency users experienced with our software.

During testing, people mostly signalled that the login page was not clear enough, that the icons on the homepage should be clickable, that it was hard to find the camera on the map that was just added and that double clicking on the timeline was not immediately clear.

This was echoed in the suggestions from users: they suggested to do testing with a separate mouse, to add a bigger login button on the home screen, and to add more usecues for, for example, double clicking on the timeline.

5.3 Conclusion and discussion

Our research question was: *can users independently use our software and how do they experience the product?* Users can independently use our product. Most of our

users could complete each task without any help, and the help we gave to users was in minor doses. This conclusion is echoed in the user experience of our software. Most users were on the 'independent' side of the likert scale for independency. Users rated their experience with our software as quite cheerful and thought it was neutrally active (6).

The limitations to this research are, of course, that mood can be determined by many other factors; we cannot be certain that our software influenced the cheerful mood. Another limitation is that, with our research, we were always present. Users might feel differently about their experience of the software if they were all alone, without anyone to help them. Right now, we were there to help if something went wrong, which could give an extra sense of security.

From these user tests, we have made the following recommendations:

- Add a login button to the home screen.
- Make icons (like 'load project') clickable.
- Make Project name clickable in 'load' project screen.
- Add a camera or player to the map by clicking on the location in the map.
- Add usecues to the timeline, so it is clear to the user that they can add a cue by double clicking.

6 Evaluation

Much of our predefined features are included in the final product. There are still some features left which we would like to see in the product. These features are stated in the next chapter. The implemented features are very intuitive for users which is something we find very important. The problems found during the user tests could be taken into account to make the product even more intuitive.
From the view of the working of the product, it does the right thing. It is easy to make a script and it is very useful to see the cameras and the players in one map to easily select a camera for a certain cue. Also to control the cues during live view is made easier for the director.

7 Outlook

Looking forward to the future of this piece of software, we can see the following things happen:

- Better integration with score. The software would map the timeline to the score and a PDF could be exported where the cues are displayed in the score itself.
- A 3D environment for the map creator and selector, making vizualization even better.
- An improved live view where cameras can be automatically controlled from the live view.
- etc.

8 References

- Bradley, M.M., and, Lang, P. (1994). Measuring emotion: The self-assessment manikin and the semantic differential. Journal behavior therapy and experimental psychiatry. 25(1), 49-59.
- Carraway, E. (2016, April 27). JohnPapa Angular 1 Style Guide. Retrieved June 15, 2016, from https://github.com/johnpapa/angular-styleguide/blob/master/a1/READM
- Nielsen, J. (1993). Usability Engineering. Boston, MA: Academic Press.

Appendices

A User Test Tasks

1. Login if necessary (username *admin*, password *admin*).
2. Load a project.
3. Add a camera to the map, giving it the name "Sony 1", type "Sony" and lense type "20mm". Move the camera to the correct position.
4. Add a player to the map, giving it the name "Conductor."
5. Go to the scripting page.
6. Add a cue to the script on bar 4 with the action "Zoom."
7. Change the bar of the cue and the duration.
8. Update the action of the cue.
9. Export the script you created to PDF.

B Notes made during testing

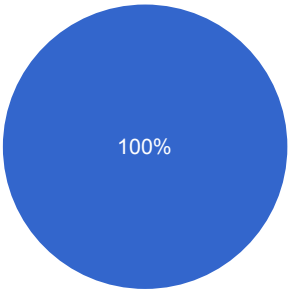
The notes made during testing, including graphs and figures representing the data that was gathered can be found on the next ten pages.

12 responses

Summary

1. Login if necessary.

1. Done



Yes	**12**	100%
No	**0**	0%

1. Time

20
15
25
40
10
30
5

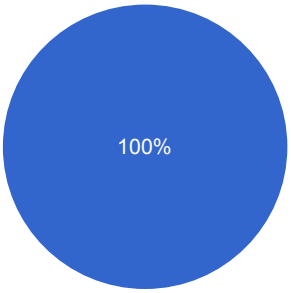
1. Notes

Lukt, duurt even om te vinden
Was hard to find login button
User did not see any big buttons, so went to the 'account' tab because of his experience with other software.
Account was found directly
Thinks it is able to go with account
easy
Wil eerst op login klikken, maar daarna op account geklikt
Gebruiker gaat eerst naar de login text in het midden van de pagina

niet zo moeilijk

2. Load a project.

2. Done



Yes	**12**	100%
No	**0**	0%

2. Time

10
20
30
25

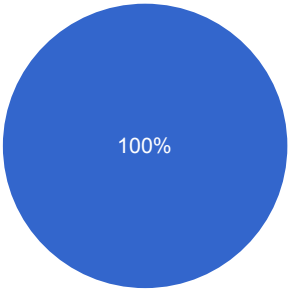
2. Notes

First clicked on 'script 1
Hard to find open button
Wanted to click on the project name. Because he did not see anything happening, he tried the other options and eventually clicked on 'open'
no problem
Clickin on icon does not work
Tried clicking on the icon
no problems
Proobeerde eerst op icon te klikken
gebruiker klikte op het project, in plaats van op open

3. Add a camera to the map.

3. Done

Yes	**12**	100%
No	**0**	0%



30
50
60
10
15

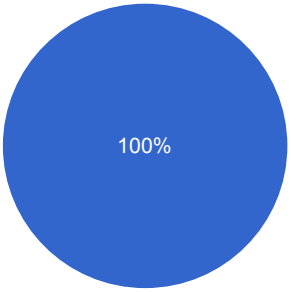
3. Notes

Was hard to see how to turn it, right clicked
The added camera was not directly found in the map, but moving was easy
X and Y values were confusing, it was not directly clear where the added camera could be found. It was not immediately clear to him that moving the cameras around was an option.
Sees empty places, thinks he can place. Add camera was easy to find. Dragging around camera was easy. Thinks the players are places where the cameras can be placed.
Finding the correct camera was hard, but dragging went easy
Found the camera quickly and moved it. Rotating was also found when prompted.
Rotation tools are very hard to see
Hij wou in eerste instantie een camera in de ruimte plaatsen door te slepen, de gebruiker kwam er pas later achter dat er informatie beschikbaar kwam wanneer hij op een camera klikte.
Vraagt zich af wat de x en de y is
Finding the add camera button was hard. Finding the rotate handle took some time too.
camera toevoegen is intuitief, niet duidelijk welke camera is toegevoegd, draai handle is niet duidelijk
Gebruiker klikt op paneel rechts om te bewerken

4. Add a player to the map.

4. Done

Yes	**12**	100%
No	**0**	0%



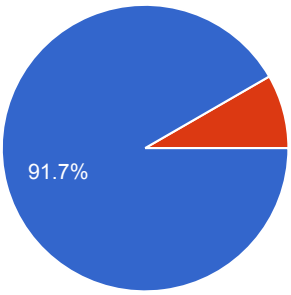
20
15
30
40

4. Notes

The added camera was not directly found in the map, but moving was easy
Easy, but the user expected to add a single player. Because of the other entries in the table, he derived that it was a player group to be added.
Knew where it was because it was at x20 y20
geen problemen, Hij vond het jammer dat hij moest scrollen om alles te zien (camera's en players)
Vraagt zich af waar de speler terecht is gekomen op de kaart, vraagt zich af waarom er geen icoontjes op de spelers zitten (anders was het wel herkenbaar geweest)
Location of the added instrument is not clear.

5. Go to the scripting page.

5. Done



Yes	**11**	91.7%
No	**1**	8.3%

5. Time

3
5
15
10

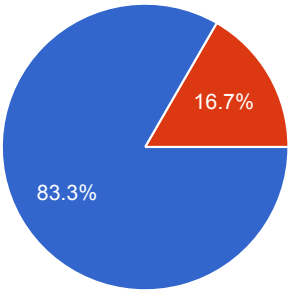
2
60

5. Notes

Was easier, because camera actions were known
No problem
dubbelklik was niet duidelijk (wel een apple gebruiker)
user went down the page first

6. Add a cue to the script.

6. Done



Yes	**10**	83.3%
No	**2**	16.7%

6. Time

10
5
40
60
15
30
20
50

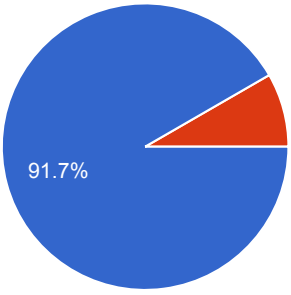
6. Notes

Double clicking was intuitive, save was hard
Double clicking was done by intuition. The user thought it was confusing to just have measures, thought it was time and expected more finegrained time control. When adding a cue, he used 00:02:00 as duration instead of an integer.
Right clicking was tried. clicking with two fingers. Finally, double clicking worked.
The user had to be told what a cue was. Double clicking was intuitive for the user

Double clicking went right away. No other things tried.
Thought double clicking already, saw nothing, because double clicking was normal. Same with calendar on Mac
Dubbelklik was niet direct duidelijk
Doubleclicking was not clear.
Dubbelklik was niet duidelijk

7. Change the bar of the cue and the duration

7. Done



Yes	**11**	91.7%
No	**1**	8.3%

7. Time

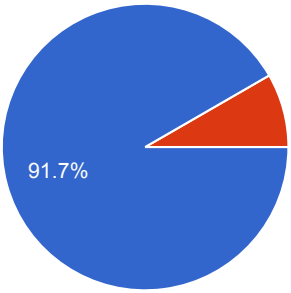
10
20
5
50

7. Notes

Double clicked
Double clicking needed help
Because the cue only moves to a bar, it was not directly clear to the user that the cue could be dragged. He did use double click, which worked, but we had to tell him that dragging around was possible.
Dragging around was hard to see, because the cue snaps to bars.
Hard to drag around, because it snaps
actie veranderen is ook niet direct duidelijk (klikken op cue)
In de scripting view veranded de muis wel als je er op staat naar een sleepding, maar je kan niet slepen. En als je dubbelklikt op de timeline zou je graag de player en de camera op de map willen selecteren.

8. Update the action of the cue.

8. Done



Yes	**11**	91.7%
No	**1**	8.3%

8. Time

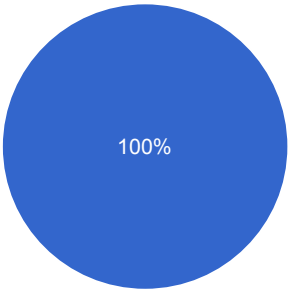
10
5
20
15
3

8. Notes

No problem
Worked
Thought 'x' was to move away from edit phase
Waar moet je heen als je nu de camera positie wil veranderen? Oh ja map. Die icoontjes zijn ook een beetje raar, je verwacht google maps. Er zit een bug in de script, als je naar home gaat en terug, is hij weg. Geen overzicht op de timeline, een scrollbar zou chill zijn.

9. Export the script you created to PDF.

9. Done



Yes	**12**	100%
No	**0**	0%

9. Time

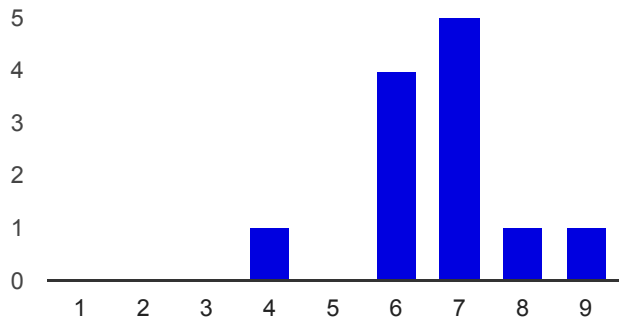
5
10
3
15

9. Notes

Had already seen the export button when the script page opened.
WERKTE NIET
WERKT NIET

Debriefing

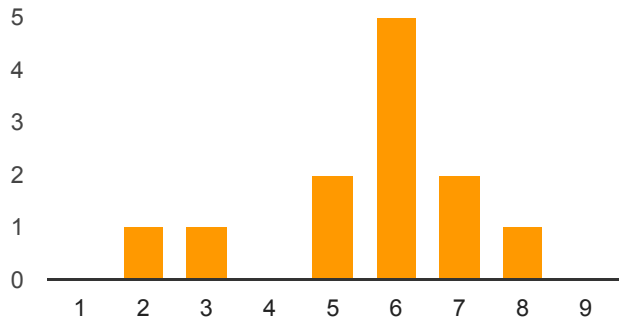
Mood



Sad:	1	**0**	0%
	2	**0**	0%
	3	**0**	0%
	4	**1**	8.3%
	5	**0**	0%
	6	**4**	33.3%
	7	**5**	41.7%
	8	**1**	8.3%
Cheerful:	9	**1**	8.3%

[Image]

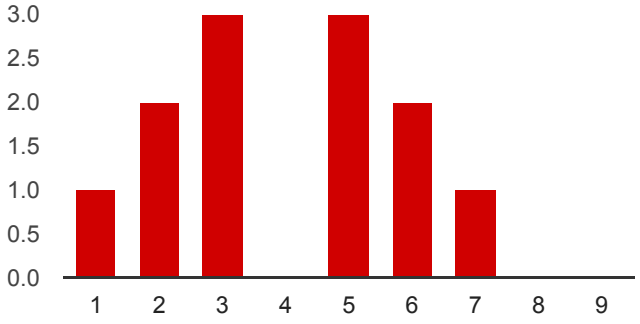
Activity



Quiet:	1	**0**	0%
	2	**1**	8.3%
	3	**1**	8.3%
	4	**0**	0%
	5	**2**	16.7%
	6	**5**	41.7%
	7	**2**	16.7%
	8	**1**	8.3%
Active:	9	**0**	0%

[Image]

Independence



Independent:	1	**1**	8.3%
	2	**2**	16.7%
	3	**3**	25%
	4	**0**	0%
	5	**3**	25%
	6	**2**	16.7%
	7	**1**	8.3%
	8	**0**	0%
Dependent:	9	**0**	0%

[Image]

Any suggestions?

Login meteen groot op scherm
Double clicking
Time was not clear to be measures
Volgende x een muis
Nieuw toegevoegde camera's of spelers een eigen kleurtje geven, of duidelijk aangeven.
Meer useques. Minder scrollen, dus kijk naar de opmaak van de pagina om een totaal overzicht weer te geven.

Number of daily responses