xBook project

Background

P2P is a company that wants to recreate the market of education in Russia and other parts of the world. For this goal they decided to change learning way, when pupils and students have paper-based or their electronic versions of the textbooks. From the point of view of P2P this approach is deprecated and textbooks should be interactive - with different elements, as exercices, mediafiles, etc., and adaptive, so that students can open them on any electronic device in any place, with or without internet.
For this purpose P2P needs three different application:

- Player - it should allow to “play” books, check theirs content, interact and so on. This part is intended for end-users: pupils, students, teachers
- Textbook editor - it should allow to edit content of the book and print them on the paper. This part is intended for the publishing houses
- Style editor - it should allow to edit how specific textbook should look like - colors, fonts, paddings and so on. This part is also intended for the publishing houses.

Our main part of the project is the style editor.

Right now P2P has their own customer for this platform - Rostelecom. It will be used for electronic courses their internal needs.

Goals

- To integrate electronic textbooks that were created on P2P’s platform into educational programs of different countries (Russia in the beggining, other counties later)
- To develop a tool for creating electronic textbooks for publishing houses
- To develop a tool that is useful and comfortable for end-users: pupils, students, even if they have vision or hearing problems

Development team

Dmitry Samoylenko Team Lead	**Evgeny Bobrov** Team member	**Ruzilya Mirgalimova** Team member

P2P team

Quick navigation

- Decision log
- How-to articles
- Meeting notes
- Meeting notes (customer)
- Meeting notes (mentors)
- Practice areas
- Process framework
- Reflection on feedback

Useful links

Link	Description
GitLab	In GitLab you can find source code for the project, see statistics for commits and development and set up Continuous Integration.
JIRA	JIRA is a place for all tasks for doing this project, iterations, bugs, and schedule. Moreover, there you can find or create own statistics based on the existing data for some reasons.
Confluence	This is a place for the documentation of the project. It doesn't matter what is it: documents, to-do lists, scripts, or retrospectives - it will be here.
SonarQube	A tool for checking quality of the application. Collects information about some quality attributes, code coverage and so on.

Tasks

- ✓ Dmitry Samoylenko check is it pos
- ✓ Ruzilya Mirgalimova move all the s
- ✓ Dmitry Samoylenko move todo: an
- ✓ Dmitry Samoylenko rebase commi
- ✓ Dmitry Samoylenko fix mail server
- ✓ Dmitry Samoylenko, Evgeny Bobro references

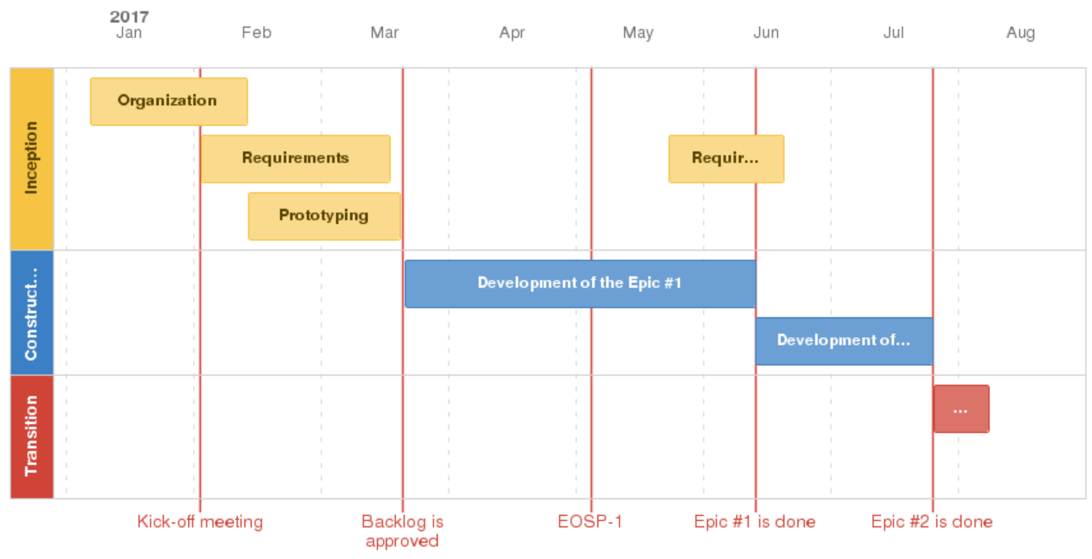
Michail Savin CEO	**Igor Akulov** CTO	**Gleb Kushedow** Lead designer	**Sergey Zverev** Lead developer

Mentor team

Panos Fitsilis Mentor	**Stanislav Litvinov** Shadow mentor

Roadmap

Here you can see actual data about previous work that was finished and estimation for the next work. Detailed plan you can see on the page of Milestone plan_.



Important points

- In the very beginning of the project customer expected that we would start to create prototypes immediately (without detailed requirements)
- For creating prototypes we had to got approval from lead designer. We had to do test exercises and only then real prototypes
- From the very beginning, customer almost couldn't explain what we need to do for the first epic - which components we need to edit and what attributes should be changeable. We had to elicit all of these points.
- Customer can't explain what we need to do for the third epic, only general thoughts
- We faced some difficulties with understanding the requirements before 21st of February, so that we had to spend extra time on that
- Diana Sunagatullina dropped from the university on 13rd of February
- We had a F2F meeting with customers (Michail Savin and Gled Kushedow) on 16th of May
- After approval for the work item list and prototypes Michail Savin told and start of the development, that we need to got approval for our prototypes and work item list one more time from Sergey Zverev
- Sergey Zverev knew almost nothing about our project: he didn't know about prototypes, approved work item list and even milestone plan
- Sometimes we couldn't communicate with Michail Savin for presenting demo because he had been in business trips
- During F2F with customer we found that second epic is much smaller than we thought
- During 5th iteration team got technical problems with implementation of the preview part

Summary

The summary of the project provided in Lessons learned page.

Additional information for topics:

- Summary and reflection for Planing provided here. Additional summary for Planning and estimation and Milestone plan_ added to documents respectively.
- Added additional summary and reflection for Tracking as well.

Practice areas

Requirements management

INVEST method

To be sure that the stories that we have written are good ones, we use decided to use six criteria: Independent, Negotiable, Valuable, Estimatable, Small (sized appropriately), and Testable, which was offered by Bill Wake in his original article "Good stories, and SMART tasks" [1]. All this criteria summarized by the acronym INVEST.
To better understand between customer, product owner and development team, how this criterias apply to our project we add table of explanation with estimation criteria.

I	Independent	How much user stories are independent or at least only loosely coupled with one another.	3 (fully independent), 2 (loosely coupled), 1 (depend on other user stories)
N	Negotiable	How much user stories are placeholders for the conversations where the details will be negotiated and not written contract in the form of an up-front requirements document.	3 (negotiable), 2 (needs to be checked), 1 (written contract)
V	Valuable	How much user stories are valuable to a customer, user, or both.	3 (essential), 2 (likely to have), 1 (inessential)

E	Estimatable	Are user stories estimatable by the team that will design, build, and test them	3 (easy to estimate), 2 (needed a prototype), 1 (hard to estimate)
S	Small	Are user stories sized appropriately for fits in one sprint?	3 (small enough), 2 (takes full effort), 1 (not fits to sprint and should be split)
T	Testable	Are user stories having good acceptance criteria (AC)?	3 (have clear AC), 2 (have vague AC), 1 (do not have AC)

After assess every user story with INVEST criteria we summarise the points for every user story and base on this numbers understand how good the story is, and is this story need any adjustment.

MoSCoW method

As prioritisation technique to reach a common understanding of importance of the tasks we choose MoSCoW method, which was developed by Dai Clegg [2]. Because it's easy to understand as for customer as for development team and this method widely use not only software development but in other areas as well. For example in management and business analysis. The idea of the method is that, the term *MoSCoW* is an acronym and describing in following way *Must have*, *Should have*, *Could have*, and *Won't have but would like.* Thus, in following table we give additional information that applicable for our product backlog and share this contest table with our customer as well.

Must have	critical to the delivery
Should have	important but not necessary for delivery
Could have	desirable but not necessary for delivery
Won’t have	least-critical, lowest-payback items

All our user stories, features and epics asses with MoSCoW method and approved by our customer.

Reference:

1. Bill Wake. "Good stories, and SMART tasks".
   Link: http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/
2. Clegg, Dai; Barker, Richard (2004-11-09). *Case Method Fast-Track: A RAD Approach*. Addison-Wesley. ISBN 978-0-201-62432-8.

Prototypes

Description

Through prototyping process I used Task-Centered User Interface Design [1], which was suggested by Lead designer from customer side Gleb Kushedow.

While we gathered requirements we incrementally understood user workflow and user tasks within it, so next step should be an artefact which will connect us and customer and we need this "something" as soon as possible. The most easiest way to do it is: create lo-fi prototypes on the paper (sketches) in order to start think about Style editor from user point of view, create user stories scenarios, draw components.

After paper sketches we should to present our ideas nice way. For this reason Lead designer Gleb asked us do it in Sketch (today popular professional digital design tool for prototyping on Mac platform (only).

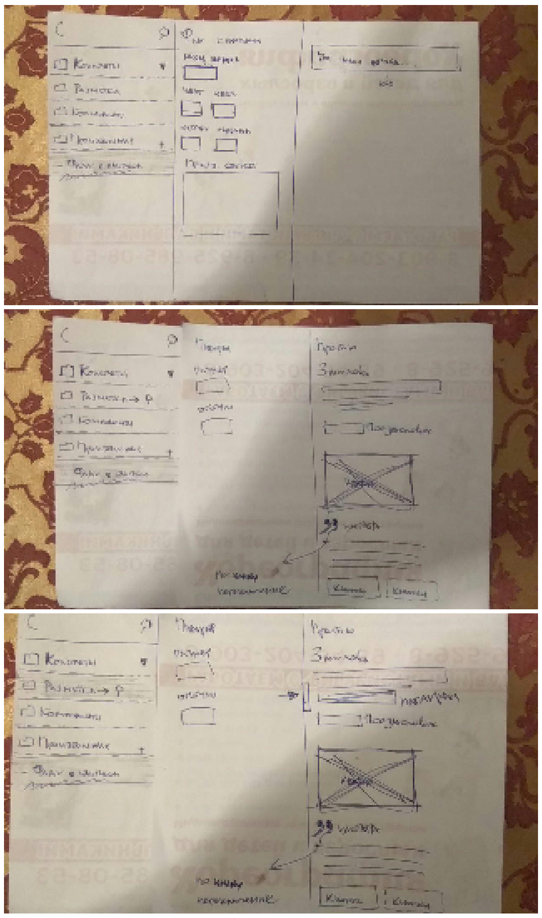
More info about Sketch: https://www.sketchapp.com

As I (Ruzilya Mirgalimova) am designer in our team wasn't familiar with it before due to using other tools and because Sketch is service to be paid. But our Product owner (Michail Savin) got a licence for me and I started doing test tasks from Gleb in order to get approval to design our Style editor.

Further there are eventual prototypes during this phase:

1. **Paper sketches (20th of February)**

Editor page:

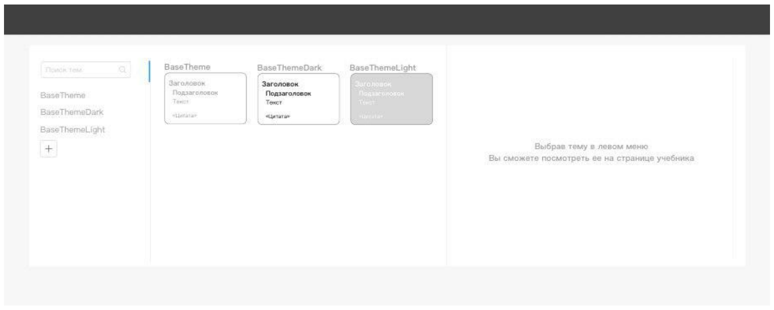


Final paper sketched from Gleb with three main parts of our Style editor: Left menu, editor area and preview.

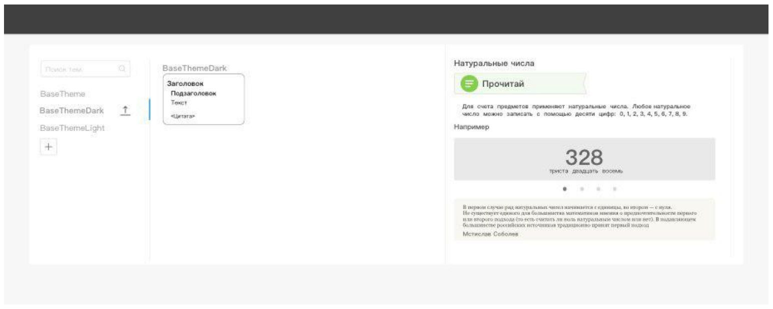
2. First prototype in Sketch (28th of February)

During this phase there is no using Ant components.

Themes page:



Выбрав тему в левом меню
Вы сможете посмотреть ее на странице учебника



Натуральные числа

Прочитай

Для счета предметов применяют натуральные числа. Любое натуральное число можно записать с помощью десяти цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

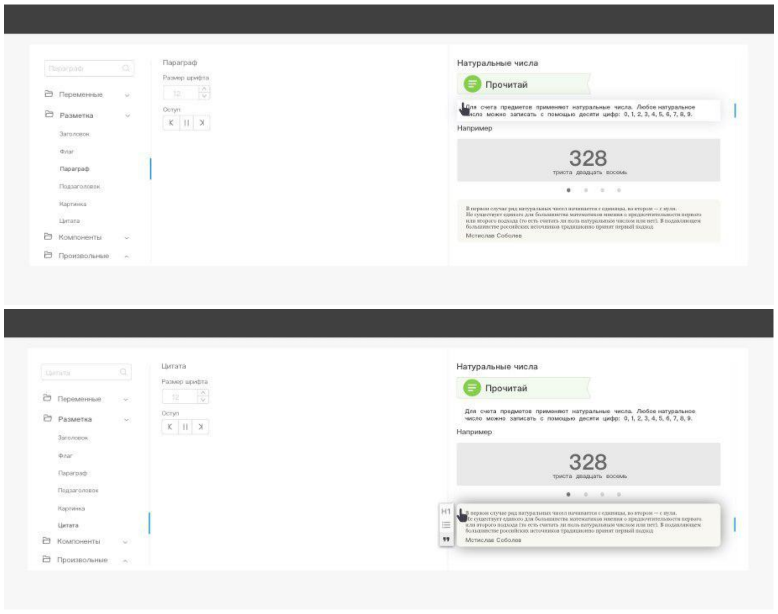
Например

328
триста двадцать восемь

В первом случае ряд натуральных чисел начинается с единицы, во втором — с нуля. Не существует единого для большинства математиков мнения о предпочтительности первого или второго подхода (то есть считать ли ноль натуральным числом или нет). В подавляющем большинстве российских источников традиционно принят первый подход

Мстислав Соболев

Editor page:



Натуральные числа

Прочитай

Для счета предметов применяют натуральные числа. Любое натуральное число можно записать с помощью десяти цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Например

328
триста двадцать восемь

В первом случае ряд натуральных чисел начинается с единицы, во втором — с нуля. Не существует единого для большинства математиков мнения о предпочтительности первого или второго подхода (то есть считать ли ноль натуральным числом или нет). В подавляющем большинстве российских источников традиционно принят первый подход

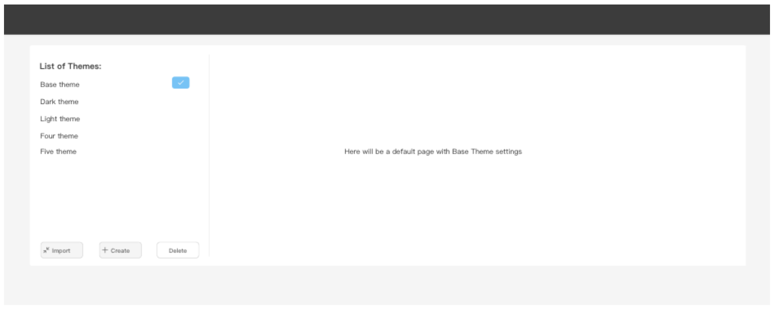
Мстислав Соболев

3. Continue prototyping (2nd of March)

There were added buttons fro import/export.

Also we think where we should set buttons. How select themes and represent it.

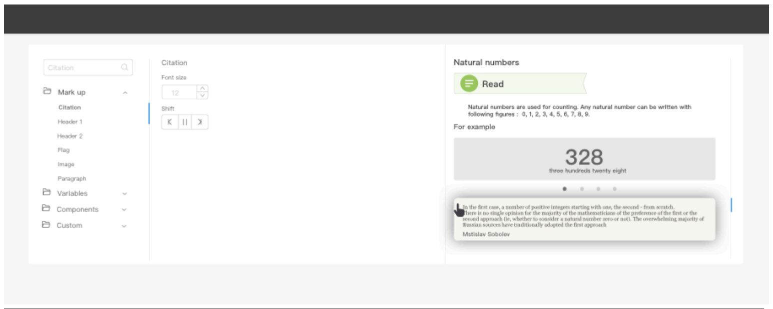
Themes page:



Here will be a default page with Base Theme settings

Editor page:

There is a disclose tree of left menu components.



Natural numbers

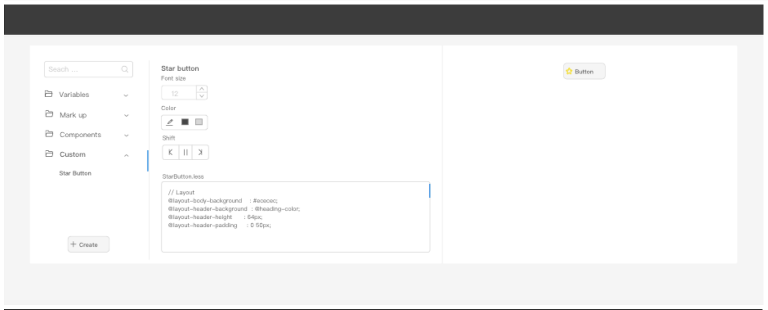
Natural numbers are used for counting. Any natural number can be written with following figures : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

For example

328
three hundreds twenty eight

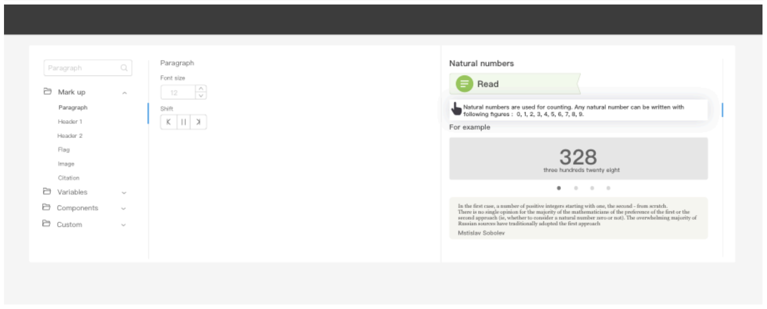
In the first case, a number of positive integers starting with one, the second - from scratch. There is no single opinion for the majority of the mathematicians of the preference of the first or the second approach (ie, whether to consider a natural number zero or not). The overwhelming majority of Russian sources have traditionally adopted the first approach

Mstislav Sobolev



StarButton.less

```
// Layout
@layout-body-background     : #ececec;
@layout-header-background   : @heading-color;
@layout-header-height       : 64px;
@layout-header-padding      : 0 50px;
```



Natural numbers

Natural numbers are used for counting. Any natural number can be written with following figures : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

For example

328
three hundreds twenty eight

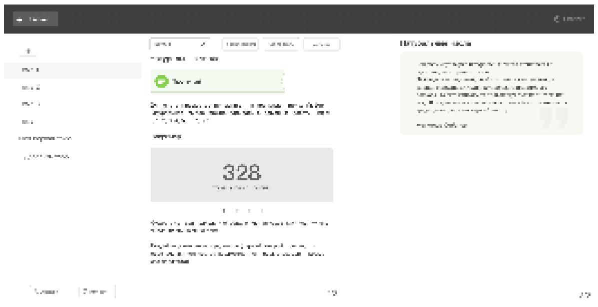
In the first case, a number of positive integers starting with one, the second - from scratch. There is no single opinion for the majority of the mathematicians of the preference of the first or the second approach (ie, whether to consider a natural number zero or not). The overwhelming majority of Russian sources have traditionally adopted the first approach

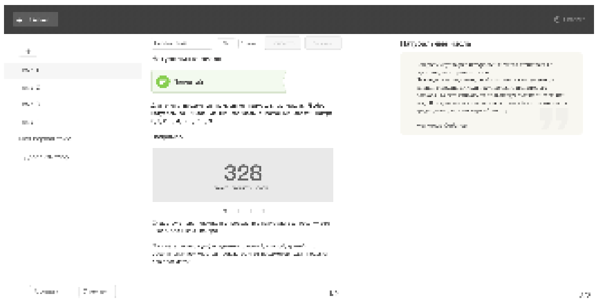
Mstislav Sobolev

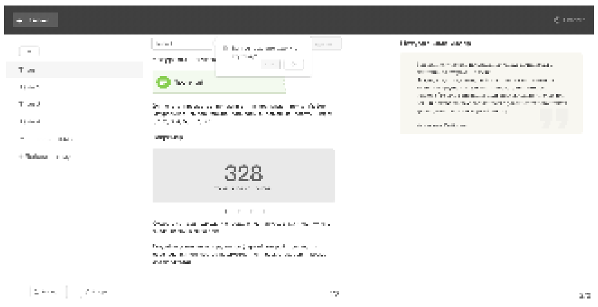
4. Final version (21st of March)

Creating set of screens for interactive prototype (in parallel with finishing to gather requirements) our three parts of applications were presented more detailed and numerous. Then screens were added to Marvel App in order to generated interactive prototype with possible user's scenarios. This final version was sent to Product owner and Lead designer for soon feedbacks.

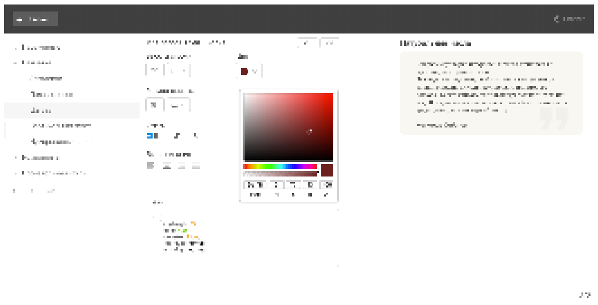
Themes page:

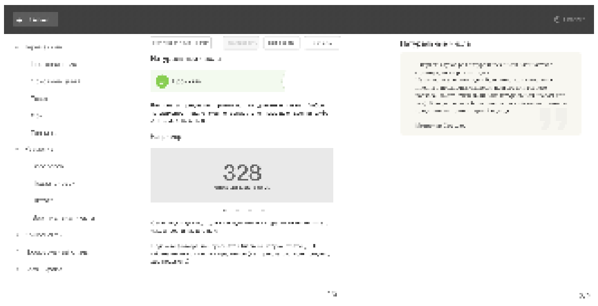






Editor page:





Reference:

1. Clayton Lewis, John Rieman. Task Centered User Interface Design. A practical introduction. 1993, 1994.
   Link: http://courses.cs.washington.edu/courses/cse440/08au/readings_files/lewis-reiman/

Work Item List

We use the term work items instead of product backlog and we depict the list differently. There are different types of things that teams work on, not just new features [1].

For example, we as a disciplined agile team members work on new functionality, fixing defects, helping each other, large technical work, regulatory documentation submissions. Also, work items are different sizes.

Add Product requirements

Title	Designer	Developers	Document owner	Document status	Epic	QA	Target release
Style Editor	Ruzilya Mirgalimova	Dmitry Samoylenko Evgeny Bobrov Ruzilya Mirgalimova	Dmitry Samoylenko	APPROVED		Dmitry Samoylenko	1.0
3. Widgets	Ruzilya	Dmitry	Dmitry			Dmitry	3.0

and components editor	Mirgalimova	Samoylenko Evgeny Bobrov Ruzilya Mirgalimova	Samoylenko	DRAFT	**STYLED-3** - Creating of new widgets or components ITERATION BACKLOG	Samoylenko	
2. Right-to-left components	Ruzilya Mirgalimova	Dmitry Samoylenko Evgeny Bobrov Ruzilya Mirgalimova	Dmitry Samoylenko	APPROVED	**STYLED-2** - RtL styles for the components of the book editor ITERATION BACKLOG	Dmitry Samoylenko	2.0
1. Style Editor	Ruzilya Mirgalimova	Dmitry Samoylenko Evgeny Bobrov Ruzilya Mirgalimova	Dmitry Samoylenko	APPROVED	**STYLED-1** - Creating styles for electronic textbooks ITERATION BACKLOG	Dmitry Samoylenko	1.0

Previous versions

Second version

Backlog v2.0.pdf

Backlog v2.5.pdf

Third version

Backlog v3.1.pdf

Backlog v3.3.pdf

Fourth version

Backlog v4... alpha.pdf

Backlog v4...0 beta.pdf

Backlog v...0 RC1.pdf

Backlog v...0 RC2.pdf

Backlog v4.0.pdf

Reference:

1. Scott W. Ambler. Going Beyond Scrum & Disciplined Agile Delivery. Disciplined Agile Consortium, 2013.
   Link: https://disciplinedagileconsortium.org/Resources/Documents/BeyondScrum.pdf
2. Scott W. Ambler. Discipline Agile Delivery. A practitioner's Guide to Agile Software Delivery in the Enterprise, 2012.

1. Style Editor

Target release	1.0
Epic	**STYLED-1** - Creating styles for electronic textbooks ITERATION BACKLOG
Document status	APPROVED
Document owner	Dmitry Samoylenko
Designer	Ruzilya Mirgalimova
Developers	Dmitry Samoylenko Evgeny Bobrov Ruzilya Mirgalimova
QA	Dmitry Samoylenko

Goals

- To provide to editors from publishing houses a tool with live preview for stylization of electronic textbooks
- To provide a backup tool for the electronic textbooks with non-standard and/or complex stylization
- To allow to use style editor for bilingual users
- To edit textbooks by using LESS code
- To provide predefined themes for the textbooks, that based on the best practices, including themes for people with disabilities

Background and strategic fit

P2P is a company that wants to recreate the market of education in Russia and other parts of the world. For this goal they decided to change learning way, when pupils and students have paper-based or their electronic versions of the textbooks. From the point of view of P2P this approach is deprecated and textbooks should be interactive - with different elements, such as exercises, mediafiles, and so on, and adaptive, so that students can open them on any electronic device in any place, with or without Internet.
For this purpose P2P needs three different application:

1. Player - allows to “play” books, check theirs content, interact and so on. This part is intended for end-users: pupils, students, teachers
2. Textbook editor - allows to edit content of the book and print them on the paper. This part is intended for the publishing houses
3. Style editor - it should allow to edit how specific textbook should look like - colors, fonts, paddings and so on. This part is also intended for the publishing houses

Here, our part is the style editor.

Assumptions

- Style Editor will be used only by desktop users
- Style Editor will be user only in Google Chrome browser or as a part of an Electron application (Chromium)

Requirements

#	JIRA issue	MoSCoW	As an/a ...	I want to...	So that...	Status	Acceptance criteria	I	N	V	E	S	T	Notes
1	**STYLED-87** - "Back to the Editor" button **DONE**	Must	User	be able to return to the editor of the textbook	I can stop editing styles and get back to the textbook	DONE	Editor is able to go back to the book from the screen with all themes	3	1	3	3	3	3	
2	**STYLED-88** - "Back to the themes" button **DONE**	Must	User	be able to return to the list of all the themes	I can stop editing theme and choose another one	DONE	Editor is able to go back to the list of all themes from the screen of styles editor	2	1	3	3	3	3	
3	**STYLED-89** - Tree of components **DONE**	Must	User	see a tree with all available components in the style editor	I will see all available components that I can edit	DONE	Editor is able to choose any component to edit from the tree of components Editor must distinguish which component is selected	3	1	3	3	3	3	
4	**STYLED-114** - Labels for default themes **DONE**	Could	User	distinguish default themes and own	I won't mix them up	DONE	Editor can distinguish where is the default theme, and where is his own	1	1	2	3	3	3	
5	**STYLED-90** - Grouping of component's attributes **ITERATION BACKLOG**	Could	User	see all the edited attributes for each components grouped by the edited value	I won't be embarrassed by complexity of the editor	NOT STARTED	All the attributes in the editor are grouped by: colors, indents, fonts, border, icon	1	1	3	3	3	3	
6	**STYLED-115** - Changing the interface language **ITERATION BACKLOG**	Could	Bilingual User	change languages of the interface in the style editor	I would be able to use the application even if I don't know some language	NOT STARTED	Editor can change language of the interface if it's available All the components of the interface should be able to translation	3	1	2	2	3	3	
7	**STYLED-98** - Variables from the LESS **ITERATION BACKLOG**	Should	Power User	see all available variables from the LESS-code	I won't make a mistake while using them	REJECTED	Editor can see the list for all the variables that were mentioned in the LESS-code	2	1	2	1	2	3	
8	**STYLED-91** - List of themes **DONE**	Must	User	see a list of all available themes	I will see all available themes that I can edit	DONE	Editor can see list of all themes	3	1	3	3	3	3	
9	**STYLED-99** - Theme renaming **DONE**	Should	User	rename theme	I can rename it if needed	DONE	Editor can rename any theme Editor can cancel renaming	2	1	2	3	3	3	
10	**STYLED-101** - Copying a theme **DONE**	Should	User	copy theme	I can create new theme that is based on the copied	DONE	Editor can create a new theme that will contain the same styles as the chosen one for copying Editor can choose name for the new theme Editor can cancel copying of the existing theme	2	1	3	3	3	3	

11	STYLED-102 - Delete a theme DONE	Should	User	delete theme	I can delete unwanted theme	DONE	Editor can delete any theme, but not the last Editor can cancel prevent deleting	2	1	3	3	3	3	
12	STYLED-116 - Importing a theme DONE	Could	User	import theme	I can restore theme	DONE	Editor can import theme to the style editor by selecting a file on the local computer Editor can cancel import	3	1	2	2	2	3	
13	STYLED-117 - Exporting a theme DONE	Could	User	export theme	I can backup theme	DONE	Editor can export theme to the local computer Editor can cancel export	3	1	2	2	2	3	
14	STYLED-118 - Preview of the theme DONE	Could	User	see a preview of the theme	I can see how it looks like	DONE	Editor can see main components of the theme Editor can see for which theme the preview is activated	2	1	2	2	2	3	
15	STYLED-92 - Start of editing of the selected theme DONE	Must	User	edit a theme	I can change how it looks like	DONE	Editor can choose a theme and then switch to the style editor	2	1	3	3	3	3	
16	STYLED-119 - Pre-built themes DONE	Could	User	have pre-built themes	I can see best practices for theme development	DONE	If textbook doesn't have any theme, then pre-built themes appear	1	1	2	2	2	3	
17	STYLED-103 - Discarding of all the changes ITERATION BACKLOG	Should	User	discard all changes	I will prevent unwished changes	IN PROGRESS	Editor can discard all changes that were made since the opening	2	1	3	2	2	3	
18	STYLED-120 - Click on the preview and immediate editing ITERATION BACKLOG	Could	User	click on any component in preview in style editor mode and start to edit it	I can choose and edit components faster	NOT STARTED	Editor can click on any component in preview and switch to it editing	2	1	2	2	3	3	
19	STYLED-104 - Undo & redo DONE	Should	User	undo or redo my changes	I will prevent unwished changes	DONE	Editor can undo any change or redo previously canceled change	2	1	3	2	2	3	
20	STYLED-93 - Theme applying DONE	Must	User	apply selected theme to the textbook	the theme will be applied to the textbook	DONE	Editor can push the button "Apply" and selected theme will be applied to the edited textbook	2	1	3	1	2	3	
21	STYLED-105 - Creating the new theme DONE	Should	User	create a new theme	the new default theme will be created	DONE	Editor can push the button "New theme" and new theme will be created based on the default pre-built theme	3	1	3	3	3	3	
22	STYLED-124 - Search for the themes ITERATION BACKLOG	Won't	User	find a theme by the it's name	I would be able to find any theme faster	REJECTED	Editor type the name of the theme and that theme appears in the list of all themes	3	1	1	3	3	3	
23	STYLED-121 - Spinner during saving of the theme, import or export ITERATION BACKLOG	Could	User	see a spinner when changes are being saved	I will know, when changes will be saved	REJECTED	In the event of changing some attribute of any component, editor will see the spinner until changes won't be saved	2	1	1	3	3	3	

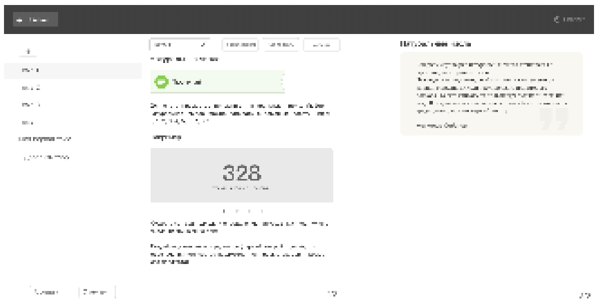
24	**STYLED-94** - Editing of the variables **DONE**	Must	User	customise values of Ant Design variables	I can control the styles of the book	DONE	Editor should be able to edit values of the variables	2	1	3	1	2	3	
25	**STYLED-95** - Preview for variables **DONE**	Must	User	see changes for the variables, that were applied to the book in preview	I can notice an error or finish editing	DONE	Editor can see what were changed in the preview	1	1	3	1	2	3	
26	**STYLED-96** - Editing of the basic markup styles **DONE**	Must	User	customize properties of the text/block (basic markup)	I can control the styles of the book	DONE	Editor should be able to edit values of the text/block	2	1	3	2	2	3	
27	**STYLED-97** - Preview for basic markup **DONE**	Must	User	see changes for the markup applied to the book in preview	I can notice an error or finish editing	DONE	Editor can see what were changed in the preview	1	1	3	2	2	3	
28	**STYLED-106** - Editing of the components **DONE**	Should	User	customize properties of the components	I can control the styles of the book	NOT STARTED	Editor should be able to edit values of the components	2	1	3	2	2	3	
29		Should	User	see changes applied to the component in the preview	I can notice an error or finish editing	NOT STARTED	Editor can see what were changed in the preview	1	1	3	2	2	3	
30	**STYLED-108** - Creating the own style **ITERATION BACKLOG**	Should	User	create own style	I can work with own text styles in book editor	NOT STARTED	Editor can create own styles and they will be shown in the tree view of components Editor should be able to name own style	3	1	3	3	3	3	
31	**STYLED-109** - Renaming the own style **ITERATION BACKLOG**	Should	User	rename the own style	I can fix a mistake in the name of the theme or change the name if needed	NOT STARTED	Editor should be able to rename own style Editor can cancel renaming	1	1	3	3	3	3	
32	**STYLED-110** - Deleting the own style **ITERATION BACKLOG**	Should	User	delete own style	I can delete unnecessary style	NOT STARTED	Editor can delete any style Editor can cancel deleting	1	1	3	3	3	3	
33	**STYLED-111** - Editing attributes for the own style **ITERATION BACKLOG**	Should	User	customize properties of the text/block	I can control the styles of the book	NOT STARTED	Editor should be able to edit values of the own style	1	1	3	2	2	3	
34	**STYLED-112** - Copying the own style **ITERATION BACKLOG**	Should	User	copy existed style and create the new one based on it	I can accelerate the speed of the editing	NOT STARTED	Editor can copy existed style (own or any other) and then the new one appears with the same attributes and values for them Editor should be able to set the name for the new style Editor should be able to cancel copying	1	1	3	3	3	3	
35	**STYLED-113** - Preview for own style **ITERATION BACKLOG**	Should	User	see changes applied to the text/block in the preview	I can notice an error or finish editing	NOT STARTED	Editor can see what were changed in the preview	1	1	3	2	2	3	
36	**STYLED-122** - Validation of the LESS **ITERATION BACKLOG**	Could	Power User	see, is inputted LESS-code correct or not	I won't save uncompilable code	REJECTED	Editor can't save wrong (that can't be compiled) LESS-code	2	1	2	2	2	3	

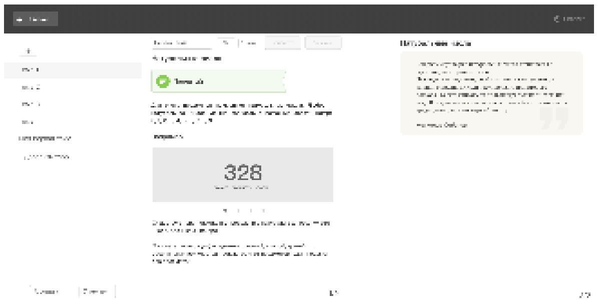
37	STYLED-12 3 - LESS highlighter ITERATION BACKLOG	Could	Power User	see highlighted LESS-code	I can distinguish variables, attributes, values, and selectors	REJECTED	Editor is able to distinguish variables, attributes, values and selectors in LESS-code	2	1	2	2	2	3	
38		Could	Bilingual User	see the applications on Russian and English as well	I can change application language	NOT STARTED	Editor can select the language of the application and it will be changed immediately	3	2	2	2	2	3	
39		Could	User	input range for component's attributes and see several results in preview	I can see components with several styles and select the most appropriate	REJECTED	Editor can enter range of values for numeral values (like margins) and specific array of values for text data (like fonts) and see in the preview components with random values from those ranges	1	2	2	2	2	3	
40		Could	Power User	type the LESS-code	I can apply even more styles to the textbook than style editor allows me	REJECTED	Editor can type LESS-code and it will be applied to the textbook	3	2	2	1	2	2	
41		Should	User	read help information	I can learn how to use the application faster	IN PROGRESS	Editor can click on the button "Help" and new page with all of this information will be opened	3	2	2	1	2	3	

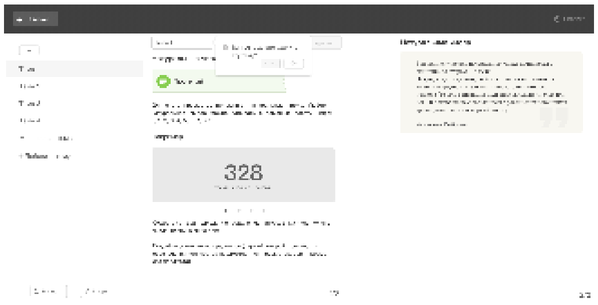
User interaction and design

Marvel live prototype: https://marvelapp.com/3331c53

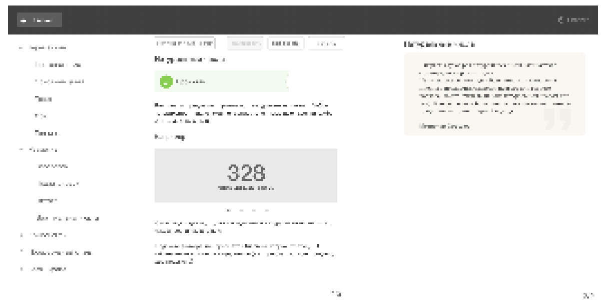
Themes page:

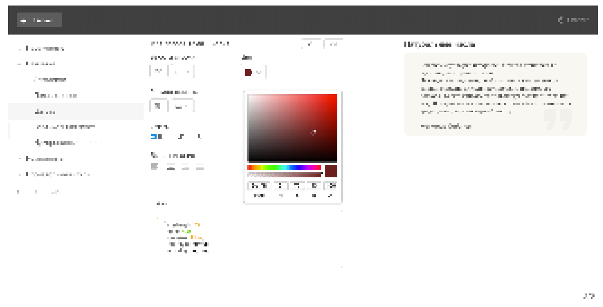






Editor page:





2. Right-to-left components

Target release	2.0
Epic	**STYLED-2** - RtL styles for the components of the book editor ITERATION BACKLOG
Document status	APPROVED
Document owner	Dmitry Samoylenko
Designer	Ruzilya Mirgalimova
Developers	Dmitry Samoylenko Evgeny Bobrov Ruzilya Mirgalimova
QA	Dmitry Samoylenko

Goals

- To allow to create RtL components in the style editor for using them in the textbooks

Background and strategic fit

P2P wants to spread electronic textbook platform not only for the market of Russia, but for the other countries as well (including countries with right-to-left direction). So that components should be available for translation on several languages and support right-to-left direction.

Assumptions

Requirements

#	JIRA issue	MoSCoW	As an/a...	I want to...	So that...	Acceptance criteria	I	N	V	E	S	T	Notes
1		Must	Publisher	have support of RtL for basic markup elements	I will be able to provide books to new markets	Basic markup, that described in the style editor, should be adapted for RtL styles when user requires it	2	3	3	3	3	3	Main user story that should be done first
2		Should	Publisher	have support of RtL for custom components	I will be able to provide books to new markets	All custom components, that described in the style editor, should be adapted for RtL styles when user requires it	1	2	2	2	3	2	Userstory that relate to the previous one. Can be start after first will be finished. Need to clarify prioritisation of supporting custom components. (components might be added in the future).

3		Could	Publisher	create own styles with support of RtL styles	own styles will work with RtL styles	In the style editor new section should be available for the own styles, where user can input CSS code for RtL	1	2	1	2	2	2	Userstory depends on previous two. Need a prototype how the custom (own) styles will be implemented.

3. Widgets and components editor

Target release	3.0
Epic	**STYLED-3** - Creating of new widgets or components ITERATION BACKLOG
Document status	DRAFT
Document owner	Dmitry Samoylenko
Designer	Ruzilya Mirgalimova
Developers	Dmitry Samoylenko Evgeny Bobrov Ruzilya Mirgalimova
QA	Dmitry Samoylenko

Goals

- To create a tool for editing and creating new widgets and components for the textbooks
- To create a tool that will satisfy any desired functionality from an author of the textbook

Background and strategic fit

P2P can't predict all the desires of the authors of the textbooks so that there should be a tool for creating new widgets and components, such as new exercises or support of media files.

Assumptions

Requirements

#	JIRA issue	MoSCoW	As an/a...	I want to...	So that...	Acceptance criteria	I	N	V	E	S	T	Notes
1													
2													

Risk management

Description

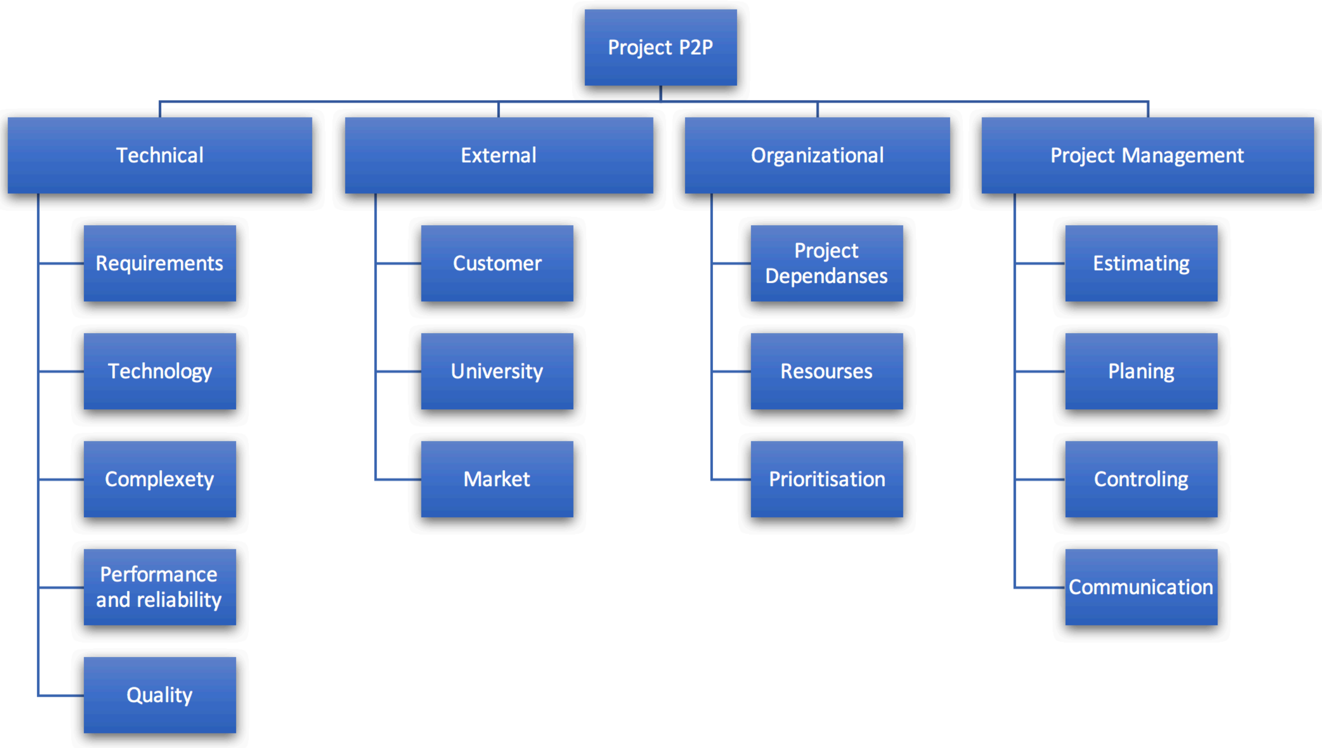
To provide quality risk management our team decided to use Boehm’s Risk Management Taxonomy structure [1] and PMBOK tactics [2] to assess risk and control them during the project.

Risk Assessment

To assess risk we decided to use the following approach. Identify the risk first, according to the common Risk Breakdown Structure (RBS), then analyze risk through quantitative and qualitative risk analysis technique, and finally prioritize them.

Risk Identification

To identify risk in all areas, we use RBS with categories and sub-categories that applicable for our project. After identifying this category we create RBS and base on this structure create a risk list. We also use standard check lists with most probable risks (from [3] and [4]) that might occur in the development teams.



According to the RBS our team identify following risks:

Technical category

Requirements: There can be a situation when the customer will change something in the original application so that out part will have to be changed also, but it will lead to requirement creep.

Requirements: There is a possibility that team won't understand what the customer want correctly, so that team will spend some time for doing wrong things.

Technology: Customer is not sure in the final stack of technologies and he can change it at any time. So that any change in the stack will lead to new learning for the team.

External category

Customer: There is a possibility to stuck in communication chain during requirements elicitation phase. E.g. one stakeholder has approved a prototype, and the other one rejected.

Customer: Customer's team is distributed and the industrial project team has not direct access (F2F) to them. Thus, the team has a possibility to fail communication part due to distributed team from the customer side.

Customer: Stakeholder can have no idea about specialization of the development team so that they will be expecting the high-quality product as fast as possible.

University: Due to the high workload at the university (additional courses), the development team can burn out and lose motivation during doing the project.

University: The customer focus on the result but the university on the development process and proper documentation. Thus, the team has a possibility to fail in following the chosen methodology, so the team has to spend additional time on some activities that might be redundant to the customer due to the scope of the project but important to educational aspect.

Customer: Customer can delegate some tasks to his own employees because it can be faster or better. Therefore, the customer can lose motivation in doing the colaborative project with the team.

Market: The project has competitors as inside as outside the country. If the new rival product hit the market earlier than the customer might stop development.

Organisational category

Project dependance: during the development phase, some requirements may vary, so that initial architecture can be useless, this might lead to lacks flexibility In architecture.

Recourses: the Industrial project is a part of studying in the university so that if someone will drop there is no need to continue the work with the project. Thus, team members has a possibility to stop doing the project due to the drop from the university

Project Management Category

Planing: Development team has a possibility to skip some important deadlines or spent much more time on development activities due to the complexity of learning new technologies. Thus, learning curves lead to delays and overrun.

Communication: Team members have never worked together so that they almost don't know each other. Because of that, the team may experience internal communication problems due to some reasons periodically.

Estimating: Due to the fact that most of the team members don't know the stack of technologies, and they do not have cooperative products, the estimation of the scope might be counted wrong.

Planing: Because of the impact from university and customer there are a lot of tasks that have to be done synchronically. Thus, it might be difficult to plan all activities.

Controlling: All team members are equal and might have a different view on the project success. Thus it might be difficult for chosen by the team project manager to control the team.

Risk Analysis

To analyze the risks, we add to every risk an ID and break risks description to following structure: risk description, statement, and context of the risk, a condition when risk might occur and potential consequence. Based on this structure we create a risk inventory table:

ID	Description	Statement	Context	Condition	Consequence
R-02	Communication chain	There is a possibility to stuck in communication chain during requirements elicitation phase. E.g. one stakeholder has approved a prototype, and the other one rejected	Stakeholders have different view on a final product	If different stakeholders will want products with different functionality	Team won't be able to start the development phase without more or less clear requirements
R-03	Architecture lacks flexibility	During development phase some requirements may vary, so that initial architecture can be useless	Team almost has not experience in creating architectures	If requirements will change during development	Architecture won't be able to fit all the needed functionality
R-04	Project team misunderstand requirements	There is a possibility that team won't understand what the customer want correctly, so that team will spend some time for doing wrong things	Customer doesn't have SoW and all the requirements are gathered from his own words	If requirements are misunderstood	Team will lose the time on gathering requirements one more time
R-05	Learning curves lead to delays and overrun	Development team has a possibility to skip some important deadlines or spent much more time on development activities due to complexity of learning new technologies	Team has only one experienced front-end developer, other team members will have to learn new technologies	If learning will be difficult and there isn't any help from experienced developer	Team will skip deadlines and overrun activities
R-06	Low team motivation	Development team has a possibility to lose motivation during doing the project	The development team can burn out due to the high workload at the university	If team will lose motivation	Members of the team won't want to do activities
R-07	Loss of team member	Team members has a possibility to stop doing the project due to the drop from the university	Industrial project is a part of studying in the university so that if someone will drop there is no need to continue the work with project	If someone will drop	Development team will loss team member
R-08	Stakeholders have inaccurate expectations	Stakeholder can have no idea about specialization of the development team so that they will be expecting the high-quality product as fast as possible	Customer didn't know almost anything about specialization of the team members	If customer will know nothing about specialization	Customer will be for the final product as soon as possible
R-09	Failure to follow methodology	Team has a possibility to fail in following the chosen methodology	The customer focus on the result but the university on the development process and proper documentation	If team doesn't follow the chosen methodology	Team can fail some activities or spent additional time on them

R-10	Lack of experience in a distributed team	Team has a possibility to fail communication part due to distributed team from the customer side	Customer's team is distributed and industrial project team has not direct access (F2F) to them	If customer's team is distributed	Issues in communication with the customer, unidentified requirements
R-11	Stakeholders becomes disengaged	Customer can lose motivation in doing project or delegate the same tasks to his own team	Customer can delegate some tasks to his own employees because it can be faster or better	If customer will be disengaged	Customer will be not motivated
R-12	Internal communication issues	The team may experience internal communication problems due to some reasons periodically	Team members have never worked together so that they almost don't know each other	If communication issues will happen	Team will lose a motivation
R-13	Planing issues	Not provide time for all activities	A lot of tasks that have to be done synchronically, because of the impact from university and customer	If planing issues will happen	Team can fail some activities or not have time to complete planning project requirements
R-14	Change of stack of technologies	Customer is not sure in final stack of technologies and he can change it in any time	Team members have almost no experience in developing and they learned new technologies. So that any change in stack will lead to new learning	If customer decided to change stack of technologies	Team will have to learn new stack
R-15	Requirement creep	There can be a situation when customer will change something in the original application, so that out part will have to be changed also, but it will lead to requirement creep	Customers had poor requirements and din't know, what exactly they want.	If customer will desire to add additional functionality or change too much	Team will not be on time
R-16	Estimation issues	The estimation of the project scope requirements might be counted wrong	Most of the team members don't know the stack of technologies, and they do not have cooperative products	If estimation issues occurs	Team, will fail to complete project requirements in provided time frame.
R-17	Uncontrolled team	All team members are equal and might have different view on the project success.	Might be difficult for chosen by the team project manager to control the team	If team members won't provide the task that they have to do	Team, might fail to complete project requirements in provided time frame or fail some activities that required by university.
R-18	Closing the project by customer	The customer might stop development.	The project has competitors as inside as outside the country, that working on the same directions.	If the new rival product hit the market earlier than the customer	Team will loose customer and have to provide the project without future implementation.
R-19	Absence a team member	The team member might be unavailable or do not have possibility to spend appropriate amount of time for the project	There can be circumstances when team member can not work required amount of time	It the team member get sick, have to participate (or preparing) to the interview, start working early etc.	The team member will not have enough time to compensate his/her absence time, thus the project will not be implemented in time/scope frame.
R-20	Customer will not test the final version of the product	Customer will not have resources to test the final version of the product.	Customer's programmers have a heavy schedule to release the main part of the product. Thus, there is a chance that they will not have time to integrate our part to the main product and test thoroughly.	If the customer will not test the integration between our product and main product.	It will be difficult to evaluate results of our project, and provide final statement for the end of semester presentation.

Risk Prioritisation

To prioritise all the risk we assess the impact and probability (following by table 2) and risk exposure according to the matrix provided in table 4. We also and possible time frame when risk might happened.

Probability	Description
Frequent	Not surprised and might occur several times
Probable	To be expected might occur repeatedly
Occasional	Might occur some time

Severity	Consequence
Catastrophic	Fall to finish project
Critical	Fall to complete significant project requirements
Serious	Fall to complete planned project requirements, fall to get expectation grade

Remote	Unlikely thought conceivable
Improbable	Unlikely, probability close to zero.

Minor	Required extra work or minor slip in schedule
Negligible	Negligible impact on the project

Table 2: Scale of Probability and Severity that applicable to P2P project.

	Frequent	Probable	Occasional	Remote	Improbable
Catastrophic	**Intolerable**	**Intolerable**	**Intolerable**	**High**	**Medium**
Critical	**Intolerable**	**Intolerable**	**High**	**Medium**	**Low**
Serious	**High**	**High**	**Medium**	**Low**	Tolerable
Minor	**Medium**	**Medium**	**Low**	Tolerable	Tolerable
Negligible	**Medium**	**Low**	Tolerable	Tolerable	Tolerable

Table 3: Risk matrix.

Based on this activities we enhanced risk inventory table and provide according to the exposure status.

ID	Description	Impact	Probability	Time frame	Exposure
R-15	Requirement creep	**Critical**	**Probable**	Development phase	**Intolerable**
R-16	Estimation issues	**Critical**	**Probable**	Fist half of the development phase	**Intolerable**
R-19	Absence of team-member	**Critical**	**Probable**	End of development phase	**Intolerable**
R-14	Change of stack of technologies	**Critical**	**Occasional**	All the time	**High**
R-07	Loss of team member	**Catastrophic**	**Remote**	All the time	**High**
R-02	Communication chain	**Serious**	**Frequent**	All the time	**High**
R-05	Learning curves lead to delays and overrun	**Serious**	**Frequent**	Development phase	**High**
R-20	Customer will not test the final version of the product	**Serious**	**Probable**	End of the project	**High**
R-10	Lack of experience in a distributed team	**Serious**	**Probable**	All the time	**High**
R-11	Stakeholders becomes disengaged	**Catastrophic**	Improbable	All the time	**Medium**
R-18	Closing the project by customer	**Catastrophic**	Improbable	All the time	**Medium**
R-03	Architecture lacks flexibility	**Critical**	**Remote**	Development phase	**Medium**
R-04	Project team misunderstand requirements	**Critical**	**Remote**	Requirements elicitation phase	**Medium**
R-06	Low team motivation	**Serious**	**Occasional**	All the time	**Medium**
R-17	Uncontrolled team	**Serious**	**Occasional**	All the time	**Medium**
R-13	Planing issues	**Minor**	**Probable**	Beginning of the development phase	**Medium**
R-08	Stakeholders have inaccurate expectations	**Minor**	**Occasional**	Requirements elicitation phase	**Low**
R-09	Failure to follow methodology	**Minor**	**Occasional**	All the time	**Low**
R-12	Internal communication issues	**Minor**	**Remote**	All the time	Tolerable

Table 4. Risk exposure table.

Risk Control

To control risks our team weight every risk and try to decided what strategy can be implemented to every risk. There are four broad strategies:

1. Avoidance – Change plans to circumvent the problem
2. Transference – Outsource risk to the third party
3. Mitigation – Reduces impact or likelihood (or both)
4. Acceptance – Take the chance of negative impact

However, due to limitation of the industrial project, we can apply transference. There is high limitation on the acceptance strategy of the risk, because our team do not have any budget and can reduce the impact of the risk by overworking to reduce impact of the risk. The team hardly can change circumstances to avoid any risk as well. Thus most of the risk get the mitigation strategy.

Risk Management planing

After risk identification we apply different strategies for them and provided steps to perform with focusing on the risk that have Intolerable and High exposure.

ID	Description	Exposure	Strategy	Steps to perform
R-15	Requirement creep	**Intolerable**	Mitigation	The team should put an effort in requirements elicitation so that risk probability will be minimised. There should be documents provided how to elicit requirements, estimate the quality of requirements. The document should be evaluated and collaborated with the customer. The customer should agree and sign the final version of requirements. If the changes in scope occur, they have to be weighted and discussed with the customer. If the changes provide a high impact on the scope than project plan should be revised.
R-16	Estimation issues	**Intolerable**	Mitigation	The team should provide metrics and apply best practice solution to increase estimation accuracy. The prototype coding should be done to assess the speed and quality of every team member. The estimations have to be revised after each iteration and milestone or when one of the risks occurs.
R-19	Absence of team-member	**Intolerable**	Acceptance	It's hard to find mitigation strategy to this type of risk, however, the steps might be related to the R-06 risk "Low team motivation", according to the fact that time might be found if there are motives to do the tasks.
R-14	Change of stack of technologies	**High**	Acceptance	Team should work on requirements as much as possible for understanding the possibility of changing. Also they should move this risk out of their responsibilities.
R-07	Loss of team member	**High**	Acceptance	Other team members can help to possible dropper to mitigate circumstances and reduce the probability of leaving the university. If a cause is not about studying in university and there are no chance to save the team member. The project manager has to negotiate with university and customer to reduce the scope of the project, revise all artifacts to the appropriate scope of the project that team might implement.
R-02	Communication chain	**High**	Mitigation	Team will organize the meeting with all the stakeholders that have different point of views on the final product
R-05	Learning curves lead to delays and overrun	**High**	Acceptance	People who have experience will help to other team members in studying. Also, team will ask for additional help or workshops from the customer side
R-20	Customer will not test the final version of the product	**High**	Acceptance	Our team can not affect to the customer. However, we can prepare additional final acceptance criteria in case if customer will not test the final version of the product. We need to aware customer and our mentors with this acceptance criteria and negotiate circumstances when this criteria will be legible and overcome the original one.
R-10	Lack of experience in a distributed team	**High**	Mitigation	In case of troubles development team should organize meeting with all concerned parties
R-11	Stakeholders becomes disengaged	**Medium**	Mitigation	Development team should involve the customer to the development process by asking reasonable questions or by proposing additional features
R-18	Closing the project by customer	**Medium**	Acceptance	Development team, should regulate the possibility to complete the project for education purposed and perform all requirement that asked by university and according last version of product backlog. Team should consider the possibility to involve customer to assess tam work at the end of the project.
R-03	Architecture lacks flexibility	**Medium**	Mitigation	Team should make effort for requirement elicitation and even in case of changed requirements there is no need in changing architecture so much
R-04	Project team misunderstand requirements	**Medium**	Mitigation	The team should write the script during every meeting with the customer and after each meeting, they should analyze all gathered information. Also, they need to approve all the requirements by the customer

R-06	Low team motivation	**Medium**	Mitigation	There are three strategies: the first one is to give this member specific tasks that he can't fail (little by little he will return), the second one to make teambuilding event, the third one to have a conversation between PM and him and PM have to find causes of loss motivation
R-17	Uncontrolled team	**Medium**	Mitigation	There team building might be provided to the team. The project manager should consider every team goal and try to engage the team members to perform these goals with positively impact to the project. There is a list of the current task might be provided by the project manager and every team member should assign those tasks that fit most for him/her and provide estimation time when the task should be done. With a low efficiency of the individual work, the collective work should be stablished when every team member works with his/her task at the same time and same place.
R-13	Planing issues	**Medium**	Mitigation	All the possible activities should be discussed during meetings, so that team won't skip any of them
R-08	Stakeholders have inaccurate expectations	**Low**	Mitigation	Team should clarify who knows what to the customer
R-09	Failure to follow methodology	**Low**	Mitigation	At least one of the team should learn the chosen methodology thoroughly and in case of questions he should resolve this problems
R-12	Internal communication issues	Tolerable	Mitigation	There are should be planned team building events of F2F meeting with problematic team member

Table 5. Mitigation risk strategy.

Risk Resolution

During our project our team faced with following risks:

- R-07 Loss of team member
- R-14 Change of stack of technologies
- R-15 Requirement creep
- R-19 Absence of team-member

To reduce the impact of the risk we implemented following steps:

R-07 Loss of team member (overcome)

At the start of the project, one of our team members decided to leave the university. Our team has to possibility to change the decision, therefore the following mitigation strategy takes a place.

Our project manager has a Skype meeting with the product owner and explains the situation. Because, there was a requirements elicitation phase, we create the final product backlog according to the resources that we have. Thus we manage customer expectation and reduce the scope.

Next step was to talk to university staff who responsible for the industrial project and try to find the way how to manage this risk. One of the options was to add additional team member by reducing one of the teams with 5 persons, however, this proposal was rejected by the university and was not very convenient for us as well. As a design was to count this circumstance during project assessment.

Thus the mitigation strategy that was proposed was implemented and impact of the risk was reduced to minimal. However, the team still have to provide all artifacts and overwork to implement proposed task fully.

R-14 Change of stack of technologies (overcome)

Because our customer desisted to change whole project architecture and refactoring most part of the code, the design was made to change Draft.js to Slate. However, we do not start to use this technology and do not provide the time to learn this material. Thus, the impact of this change was minimal for us.

Despite the fact, that this risk does not affect our team, the chance that some of the technology might change increase. Thus our team increases the priority of that risk from remote to occasional. (this is already reflected in the table. The initial table with risk is in the attachment.)

R-15 Requirement creep (overcome)

This is one of the risks, that occurred in the first code iteration. The customer desisted to add new epic to our product backlog.

As for mitigation strategy, we trade of the impact of this adjustment. Decrease the value of this new epic, reconfigure it to the feature (a big user story). The revision of the product backlog was made. And the possibility to implement this user story rose, due to decreased scope of the second epic. We also change priorities for some of the user stories to increase chances to delivery essential parts of the project.

R-19 Absence of team-member (current)*

We face with this risk couple of times, but neglect it for most of the time. Because there always was the time to catch up and proceed and it happens not often, thus it is not affected the project a lot. However, at the end of the development stage, there is more pressure on the students. Most of the students, who not find the job yet, have to participate in a job interview. It is assumed that there is proper preparation for them. One of the team-members has to start his full-time work in the University 1.5 month before the project will finish. Nevertheless, despite the fact that conditions are serious and critical, the time always might be found with proper motivation, therefore the work in this direction should be provided.

```
*The team spot the difficulties, when it currently happens, hence it is not a risk but a problem. However,
this is a risk that the problem might grow or come up again after the problem resolved.
```

Risk Monitoring

To monitor the risk we use approach provided by PMBOK Risk management processes. Thus

- monitoring residual risks
- identifying new risks
- executing risk reduction plans
- evaluating risks effectiveness throughout the project life cycle.

All this activities provided after each iteration and milestone or when one of the risks occurs.

The new risks* were added during the project:

- R-16 Estimation issues
- R-17 Uncontrolled team
- R-18 Closing the project by customer
- R-19 Absence of team-member
- R-20 Customer will not test the final version of the product

```
*These risks already added to the table 1 and treated in the table 4 and table 5 respectively.
```

According to the timeframes some of the risk already passed. Thus we excluded the following risks:

- R-04 Project team misunderstand requirements
- R-13 Activities are missing from scope
- R-10 Lack of experience in a distributed team
- R-08 Stakeholders have inaccurate expectations
- R-02 Communication chain
- R-03 Architecture lacks flexibility
- R-11 Stakeholders becomes disengaged
- R-14 Change of stack of technologies
- R-18 Closing the project by customer

The actual table now looks as following:

ID	Description	Impact	Probability	Time frame	Exposure
R-15	Requirement creep	**Critical**	**Probable**	Development phase	**Intolerable**
R-16	Estimation issues	**Critical**	**Probable**	Development phase	**Intolerable**
R-19	Absence of team-member	**Critical**	**Probable**	End of development phase	**Intolerable**
R-05	Learning curves lead to delays and overrun	**Serious**	**Frequent**	Development phase	**High**
R-07	Loss of team member	**Catastrophic**	Improbable	All the time	**Medium**
R-06	Low team motivation	**Serious**	**Occasional**	All the time	**Medium**
R-17	Uncontrolled team	**Serious**	**Occasional**	All the time	**Medium**
R-09	Failure to follow methodology	**Minor**	**Occasional**	All the time	**Low**
R-12	Internal communication issues	**Minor**	**Remote**	All the time	Tolerable

Table 6. Current Risk exposure table.

Summary

We do not put enough effort to the risk management at the start of the project. Thus, when some critical risks happened we were not prepared for them and the process suffers significantly.

After MOSP the risk management was improved to a great extent and manages constantly.

Most catastrophic, critical and serious risk was updated and provide mitigation strategy. Thus we end the project with additional critical risks that happen but with customer satisfaction.

At the end of the project the risk occurred:

- R-16 Estimation issues

The risk resolution is defined in the summary and reflection part of the Milestone plan_ documentation.

Other risks that provided in Table 6, continued until the end of the project but were not happens.

Reflection

Focus on the risk strategy helps our team to reach the very important goal: customer satisfaction.

Despite the difficult circumstances we manage critical and catastrophic risks to prevent them from occurs. We also provide effective mitigation strategy that allows us not to panic when some risks happened and follow clear steps to reduce harm effect and keep stay on track.

In the future projects, we will focus on the risks from very begging of the projects and put more effort not only to provide mitigation strategy but proactive strategy as well in the case to prevent risk happening.

Previous version

Risks v2.1.pdf

Reference:

1. Barry W Boehm, Software Risk Management, IEEE Computer Society Press, Piscataway, NJ, USA, 1993.
2. Project Management Body of Knowledge**.** PMBOK® Guide and Standards.
   Link: https://www.pmi.org/pmbok-guide-standards
3. IT Pro Portal (international media group and leading digital publisher).
   Link: www.itproportal.com
4. John Spacey. Business Encyclopedia.
   Link: www.simplicable.com

- Boehm, B. *IEEE Tutorial on Software Risk Management*. Piscataway, NJ: IEEE Computer Society Press, 1989.
- Charette, R. *Software Engineering Risk Analysis and Management*. New York, NY: McGraw-Hill, 1989.
- Williams, R.; Pandelios, G.; & Behrens, S.
- Dorofee, A.; Walker, J.; Gluch, D.; Higuera, R.; Murphy, R.; Walker, J.; & Williams, R. *Team Risk Management Guidebook*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1994.
  Link: Software Risk Evaluation (SRE) Method Description (Version 2.0) & SRE Team Members Notebook (Version 2.0) (CMU/SEI-99-TR-029, ADA001008).
- Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999.
- Keuffel, W. "Planning for and Mitigating Risk." *Software Development 7*, 9 (September 1999).

Planning

To make reasonable estimates of resources, cost, and schedule we need to provide a framework of project planing process and follow to it during all time working with that project.

According to the Roger Pressman [1], there is a set of tasks for project planning:

1. Establish project scope.
2. Determine feasibility.
3. Analyse risks.
4. Define required resources.
    a. Determine required human resources.
    b. Define reusable software resources.
    c. Identify environmental resources.
5. Estimate cost and effort
    a. Decompose the problem.
    b. Develop two or more estimates using size, function points, process tasks, or use cases.
    c. Reconcile the estimates.
6. Develop a project schedule.
    a. Establish a meaningful task set.
    b. Define a task network.
    c. Use scheduling tools to develop a time-line chart.
    d. Define schedule tracking mechanisms.

To stick with this planing process, we create following activities:

Establish project scope.

To observe all scope of the project we create a work item list according Discipline Agile process framework. This work item list based on user stories that were developed and evaluated with our customer. The work item list is creating by iterative approach. Thus, the resent tasks are explained more thoroughly and estimated in a more precise way. All clarifications and changing in the work item list consider in the team and agreeing with the customer.

Determine **feasibility**.

To determine feasibility, we prioritise all user stories in the work item list and evaluate them by INVEST criteria. We also adding acceptance criteria to each of the user story to have clear understanding if the user story fully finished and ready to deploy.

Analyse risks.

To provide quality risk management our team decided to use Boehm’s Risk Management Taxonomy structure and PMBOOK tactics to assess risk and control them during the project. To identify risk in all areas, we use RBS with categories and sub-categories that applicable for our project. After identifying this category we create RBS and base on this structure create a risk list. We also use standard check lists with most probable risks that might occur in the development teams. Thus, we created risk inventory table, based on the risk matrix we structure our risk in the risk exposure table. To control the risks, mitigation risk strategy table was added.

Define required resources

To estimate overall project based on our time constraints and resources that we have, we create work breakdown structure and milestone plan. We also use it to monitor deadlines, identify potential bottlenecks and monitor dates that important for our project.

Estimate cost and effort

To estimate cost and effort we use story points for our user stories. To do that we use Planning Poker technique. This approach helps to avoid the influence of the other participants. Thus it helps to think independently on each task and propose numbers that address to particular task simultaneously with justification of choice that was made. Additional benefit of this technique is that we learn from each other how to predict more precisely and accurate, think about task from different points of view and consider additional effects that might occur during task development.

Our team holds a Planning Poker session after an initial work item list was written. We use this session to create initial estimates in scoping or sizing the project. Due to the fact that our work item list evaluate iteratively thorough development time we continue use this technique for each planning sessions once per iteration.

Develop a project schedule

For project schedule and project monitoring we use the Atlassian JIRA tool that address to project and task management solution. We deciding to use this software, because it focus on flexible planning, accurate estimations and value-driven prioritisation. There are also flexible reports that helps to be on track.

Summary and Reflection

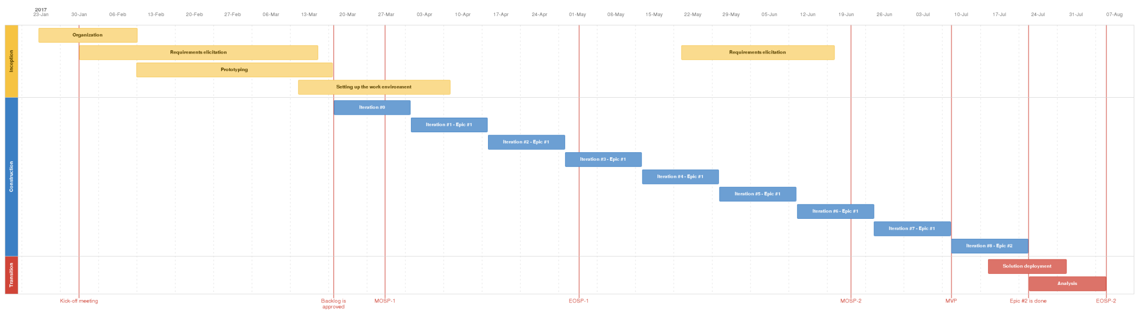
We decided that it's better to provide summary and reflection for Milestone plan_ and Planning and estimation process separately and focus on this two parts. Thus you may find this information in the relevant documents.

Reference:

1. R. S. Pressman, Software Engineering: a particular approach, New York, NY: The McGraw-Hill Companies, Inc, 2010.

   Diaz, Oscar; Trujillo, Salvador; & Anfurrutia, Felipe I. "Supporting Production Strategies as Refinements of the Production Process," 210-221. *Proceedings of the 9th International Software Product Lines Conference (SPLC 2005)*. Rennes, France, September 26-29, 2005. New York, NY: Springer, 2005.
   Jones, L. & Northrop, L. "The Establishing Phase: Planning for Successful Improvement." *Software Process: Improvement and Practice 2*, 1 (March 1996).
   Keuffel, W. "Planning for and Mitigating Risk." *Software Development 7*, 9 (September 1999).
   Ryans, A.; More, R.; Barclay, D.; & Deutscher, T. *Winning Market Leadership: Strategic Market Planning for Technology-Driven Businesses*. New York, NY: John Wiley & Sons, 2000.

Milestone plan



Summary

Due to the fact that, some critical risk occurred, our team need to revise milestone plan several times, most important revision with changes times was:

1. One of the team members left the team.
2. Customer refactor whole parent project with new architecture implementations
3. Customer change priorities of the project.
4. The customer does not provide essential information and source code of the components that should be changed.
5. Two of the three team members start to work in university and ought to pass 48 hours mandatory workshop.

Due to these factors:

- Epic 3, was rejected in the first part of the project.
- The scope of the epic 2 was shrink in the second part of the project.
- Epic 2 was eliminated from the acceptance criteria due to lack of the time.

However, an essential functional of the final working product was shipped to the customer for the test.

The functional cover 80% of the planned and might be used for the most task.

The milestone plan was revised with the customer and all revision was agreed by him.

Nevertheless, the product that was delivered satisfy the customer needs.

Reflection

The team agreed that more effort might be done to achieve a better result. However, the team thinks that the requirements were elicited and prioritized in a proper manner, that allows reaching the high level of customer satisfaction.

The team provide good managing customer expectations skills and find the proper way to negotiate with the customer.

This skills might help to conduct negotiations in future.

In advanced, we understand that we have to:

- increase management skills to motivate team members
- estimate tasks and predict future outcomes in a more effective way

- revise milestone plan and see the outcomes before the deadlines.

There are other factors that might affect on the planning part of the project, but team chooses the significant ones.

Planning and estimation

Description

We use planning poker for planning our sprints by using Story Points technique. For this we use Fibonacci numbers:

- 0 - this User Story is implemented already or too small
- 1, 2, 3 - small User Stories, can be done easily
- 5, 8 - medium and big User Stories
- 13 - really big User Story
- 21, 34 - estimations for too big User Stories, that should be divided
- 55 - in most cases this is feature or small epic
- 89 - epic

We didn't do Planning Poker for the first iteration because we didn't have any data for even approximate estimations.

Iterations

User Story	JIRA issue	Estimation	Time spent	Ration 1sp~h	Ratio	Description
#1	**STYLED-87** - "Back to the Editor" button DONE	2 sp	7h	3.5	correct estimation	Small, initial task, estimation was correct
#2	**STYLED-88** - "Back to the themes" button DONE	1 sp	3h	3.0	correct estimation	Small, initial task, estimation was correct
#3	**STYLED-89** - Tree of components DONE	5 sp	16h	3.2	correct estimation	The task was planned to do in the 1 iteration, how ever we cannot create this task without model. We decided to shift this task when model will be fully created. The task was doe in 6 iteration. Estimation and time spent correlates as planned.
#4	**STYLED-114** - Labels for default themes DONE	1 sp	2.5h	2.5	correct estimation	Small, initial task, estimation was correct
#8	**STYLED-91** - List of themes DONE	13 sp	48h	3.7	correct estimation	Large task took almost all effort in 1 iteration.
#9	**STYLED-99** - Theme renaming DONE	3 sp (+13 extra sp)	56h	18.7 (3.5)	underestimate (bad task formulation)	The task was actually spliced on 2 parts. • First part was developed in 2 iteration (the basic activities). • Second part (realisation in model) was planned to be done in third. • however, due to focus on the other tasks, was moved and done in 4 iteration/
#10	**STYLED-101** - Copying a theme DONE	2 sp	7h	3.5	correct estimation	Small, initial task, estimation was correct. Was planned and done in 2 iteration
#11	**STYLED-102** - Delete a theme DONE	2 sp	5.5h	2.8	correct estimation	The task takes 2 story points how was planned. However, it was frozen on the 1st iteration and done in iteration 6.

#12	**STYLED-116** - Importing a theme **DONE**	8 sp	18h	2.25	overestimate	Because underestimate in iterations 4-5 iterations, the tasks in iteration 6 - 8 was credit with extra time For issue 12 time was overestimated, for issue 13 we finished in time.
#13	**STYLED-117** - Exporting a theme **DONE**	5 sp	18h	3.6	correct estimation	
#15	**STYLED-92** - S tart of editing of the selected theme **DONE**	5 sp	15h	3.0	correct estimation	
#16	**STYLED-119** - Pre-built themes **DONE**	8 sp	43h	5.3	underestimate	The task planned as 5 story points, but was extended to 8. However the realisation took even more time than planned/
#17	**STYLED-103** - Discarding of all the changes **ITERATION BACKLOG**	2 sp	12h	6.0	underestimate	**Not finished yet** The task was initially started in 3 iteration, but we could not finished it due to absent of customer information. Task was moved to iteration 4. But was frozen in same reason. We got the information that was needed on iteration 5, but do not have time to implement this task according priority. Thus why the task was shifted to the last iteration.
#19	**STYLED-104** - Undo & redo **DONE**	3 sp	28h	9.3	underestimate	Because we can not get information from the customer, the task was shifted from third iteration to fifth, but actually we can implement this task only in iteration 7. Most of the task took more time than planned, because the planing poker was made in 2 and 3 iteration without deep knowledge of tasks specific.
#14	**STYLED-118** - Preview of the theme **DONE**	13 sp	48h	3.7	correct estimation	
#24	**STYLED-94** - E diting of the variables **DONE**	3 sp	24h	8.0	underestimate	
#25	**STYLED-95** - P review for variables **DONE**	5 sp	28h	5.6	underestimate	
#26	**STYLED-96** - E diting of the basic markup styles **DONE**	13 sp	24h	0.5	overestimate	
#20	**STYLED-93** - T heme applying **DONE**	13 sp	45h	3.5	correct estimation	Because this task related to the other and was performed in the 6th iteration, the estimation and actual implementation was done correctly.
#21	**STYLED-105** - Creating the new theme **DONE**	3 sp	10h	3.3	correct estimation	Small, initial task, estimation was correct

#27	**STYLED-97** - P review for basic markup DONE	8 sp	42h	5.3	underestimate (bad task formulation)	The task was actually spliced on 2 parts. • First part was started in 3 iteration (the basic activities), but was frozen due to other task relation and implemented in iteration 4.. • Second part (realisation in model) was planned to be done in fifth iteration. but was frozen and shifted as well for the same reason. Model realisation was completed in 7th iteration.
#41		5 sp				**Not finished yet** We are currently work with this task
#5	**STYLED-90** - G rouping of component's attributes ITERATION BACKLOG	2 sp	-		not implemented	**Not finished** Planned for 5 iteration but, we do not have time to implement it, and shifted to the backlog due to the priority.

To decide if we cave a correct estimation we link our **story point to 3.5 hours**. From this point we can analyse if the following estimations were wrong (under or overestimated).

Summary

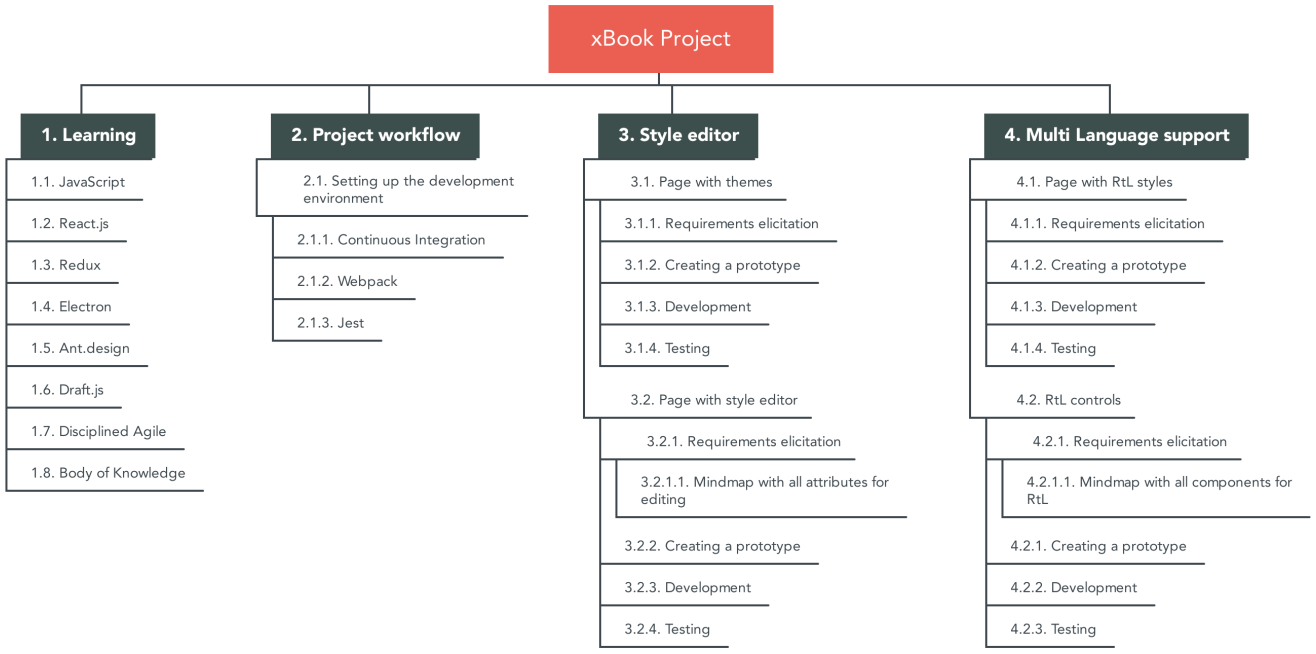
Our team does not provide enough effort to monitor and develop estimation strategy. The main reason was the lack of programming development in the current domain. However, we agreed that if we follow the strategy that was invented at the beginning we might predict outcomes and prevent risk R-16 (Estimation issues) to happen.

We also faced with the problem, when customer promised but not provide essential pieces of code that were needed to implement features. Our mistake was, to plan and then pick (f.ex iteration 3 and 4) the user stories that rely on such customer delivery. Finally, we have to postpone and freeze tasks that can not be implemented because of such reason.

Reflection

In the future projects, we all team members have to choose the task to implement and minimize issues that have to be frozen or postponed during the development phase no matter what. We have to predict possible risks not to implement the feature, thus wisely consider can we add it to the sprint or not.

WBS


xBook Project
1. Learning
1.1. JavaScript
1.2. React.js
1.3. Redux
1.4. Electron
1.5. Ant.design
1.6. Draft.js
1.7. Disciplined Agile
1.8. Body of Knowledge
2. Project workflow
2.1. Setting up the development environment
2.1.1. Continuous Integration
2.1.2. Webpack
2.1.3. Jest
3. Style editor
3.1. Page with themes
3.1.1. Requirements elicitation
3.1.2. Creating a prototype
3.1.3. Development
3.1.4. Testing
3.2. Page with style editor
3.2.1. Requirements elicitation
3.2.1.1. Mindmap with all attributes for editing
3.2.2. Creating a prototype
3.2.3. Development
3.2.4. Testing
4. Multi Language support
4.1. Page with RtL styles
4.1.1. Requirements elicitation
4.1.2. Creating a prototype
4.1.3. Development
4.1.4. Testing
4.2. RtL controls
4.2.1. Requirements elicitation
4.2.1.1. Mindmap with all components for RtL
4.2.1. Creating a prototype
4.2.2. Development
4.2.3. Testing

Previous versions

WBS v1.0.pdf

Tracking

We use different tracking strategy [1] to assess progress against the project plan and take any necessary action to maintain the schedule, such as:

- We conducted periodic project status meetings in which each team member reports progress and problems
- Determining whether formal project milestones have been accomplished by the scheduled date
- Comparing the actual start date to the planned start date for each project task in Work Item List.
- Meeting with mentors to obtain their subjective assessment of progress to date and problems on the horizon
- Using story points value analysis to assess progress quantitatively.
- Use JIRA tool conduct all changes during iteration and the project in whole.

If during the tracking activities problems were diagnosed we exercise control to reconcile them as quickly as possible. The risk table revised as well to detect if the risk were observed and res;ution strategy was prepared. If the problem is serious the project schedule can be redefined with the customer awareness.

Summary

Despite of the powerful tool such a JIRA, the tracking process in most cases managed by the people. Our team relate on the tool hopes that this will show problems and provide quality report information. However, lack of attention to the process at first interactions provide bad result that affect on the process in the future (see pic. 1). The tool that focus on the full time development wasn't the best way to conduct process with 12h workload per week without proper estimation and continuous future delivery.

Picture 1. Burn-down chart for Iteration #2.

Thus, the tracking process was conduct manually and evidence of requirements creep or estimation issues was not so obvious. Effort that was spend to the create proper reports in JIRA tool was unintended decision. However, the tool was used to create iterations, pick tasks from the backlog and conduct the process during iteration phase.

Reflection

In future work we will be consider on the process of the tracking and monitoring activities rather that rely on the systems such as JIRA or YouTrack. This tools plays more complimentary role.

Reference:

1. R. S. Pressman, Software Engineering: a particular approach, New York, NY: The McGraw-Hill Companies, Inc, 2010.

- R. Kaur, S. Singh, Dr. M. Rakshit. A Review of various Software Project Scheduling techniques: International Journal of Computer Science & Engineering Technology (IJCSET).
  Link: https://pdfs.semanticscholar.org/fdc0/7cd0724c7c791bf5436edf81b1da42ab3ce5.pdf
- S. Soroczak. D. W. McDonald. Collaborating Over Project Schedules.

  Link: http://www.pensivepuffin.com/dwmcphd/papers/Soroczak.CSCW06.Workshop.final.pdf
- Gantt, H.L. (1919) Organizing For Work. Harcourt, Brace, & Howe. New York, NY.

Iterations

TODO:

- ✓ Dmitry Samoylenko add charts when 2nd iteration will end 02 May 2017

Iteration #1

We were not able to calculate some metrics from this sprint due to moving from one JIRA server to another.

Issues

Number	JIRA issue	Summary	Priority	Status	Story Points
1	STYLED-87 - "Back to the Editor" button DONE	"Back to the Editor" button	High	DONE	2
2	STYLED-88 - "Back to the themes" button DONE	"Back to the themes" button	High	DONE	1
8	STYLED-91 - List of themes DONE	List of themes	High	DONE	13

15	**STYLED-92** - Start of editing of the selected theme **DONE**	Start of editing of the selected theme	High	DONE	5
4	**STYLED-114** - Labels for default themes **DONE**	Labels for default themes	Low	DONE	1
3	**STYLED-89** - Tree of components **DONE**	Tree of components	High	FROZEN	5
Total:					**27**

Test Coverage

File	% statements	% branch	% functions	% lines
actions.js	0	0	0	0
App.js	100	100	100	100
Content.js	66.67	50	50	66.67
Footer.js	100	100	100	100
Header.js	100	100	100	100
Sider.js	100	100	100	100
ThemesList.js	76.47	100	66.67	73.33
ComponentsContainer.js	75	100	50	75
PreviewContainer.js	83.33	100	66.67	80
ThemesContainer.js	83.33	100	66.67	80
themes.js	100	75	100	100
Total:	**63.77**	**84.21**	**55.88**	**70.18**

Code Quality

Lines of Code: 557

Reliability rating: A
Security rating: A
Maintainability: A

Duplications: 16,4% (2 blocks, 114 LoC)
Technical Debt: 2 hours 16 minutes
Code Smell: 32 blocks
Bugs: 0
Vulnerabilities: 0

Iteration #2

Issues

Number	JIRA issue	Summary	Priority	Status	Story Points
9	**STYLED-99** - Theme renaming **DONE**	Theme renaming	Medium	DONE	3

10	**STYLED-101** - Copying a theme DONE	Copying a theme	Medium	DONE	2
11	**STYLED-102** - Delete a theme DONE	Delete a theme	Medium	DONE	2
21	**STYLED-105** - Creating the new theme DONE	Creating the new theme	Medium	DONE	3
14	**STYLED-118** - Preview of the theme DONE	Preview of the theme	Low	FROZEN	13
Total:					**23**

Test Coverage

File	% statements	% branch	% functions	% lines
actions	100	100	100	100
App.js	100	100	100	100
ComponentsList.js	100	100	100	100
Content.js	22.73	23.08	40	23.81
Footer.js	100	100	100	100
Header.js	100	100	100	100
Sider.js	100	100	100	100
ThemesList.js	59.09	78.95	55.56	60
ComponentsContainer.js	75	100	50	75
ContentContainer.js	50	100	33.33	44.44
ThemesContainer.js	62.5	100	50	57.14
initialState.js	100	100	100	100
themes.js	100	85.71	100	100
Total:	**63.92**	**62.5**	**61.11**	**62.92**

Code Quality

Lines of Code: 624

Reliability rating: A
Security rating: A
Maintainability: A

Duplications: 0%
Technical Debt: 2 hours 3 minutes
Code Smell: 34 blocks
Bugs: 0
Vulnerabilities: 0

Iteration #3

Issues

Number	JIRA issue	Summary	Priority	Status	Story Points
9	**STYLED-89** - Tree of components DONE	Tree of components	Must	FROZEN	13

17	**STYLED-103** - Discarding of all the changes ITERATION BACKLOG	Discarding of all the changes	Low	NOT STARTED	2
19	**STYLED-104** - Undo & redo DONE	Undo & redo	Low	NOT STARTED	3
24	**STYLED-94** - Editing of the variables DONE	Editing of the variables	High	NOT STARTED	3
25	**STYLED-95** - Preview for variables DONE	Preview for variables	High	FROZEN	5
27	**STYLED-97** - Preview for basic markup DONE	Preview for basic markup	High	FROZEN	8
Total:					**34**

Test Coverage

File	% statements	% branch	% functions	% lines
actions	100	100	100	100
App.js	100	100	100	100
ComponentsList.js	100	100	100	100
Content.js	22.73	23.08	40	23.81
Footer.js	100	100	100	100
Header.js	100	100	100	100
Sider.js	100	100	100	100
ThemesList.js	59.09	78.95	55.56	60
ComponentsContainer.js	75	100	50	75
ContentContainer.js	50	100	33.33	44.44
ThemesContainer.js	62.5	100	50	57.14
initialState.js	100	100	100	100
themes.js	100	85.71	100	100
Total:	**63.92**	**62.5**	**61.11**	**62.92**

Code Quality

Lines of Code: 624

Reliability rating: A
Security rating: A
Maintainability: A

Duplications: 0%
Technical Debt: 2 hours 3 minutes
Code Smell: 34 blocks
Bugs: 0
Vulnerabilities: 0

Iteration #4

Issues

Number	JIRA issue	Summary	Priority	Status	Story Points

9	**STYLED-89** - Tree of components DONE	Tree of components	Must	DONE	13
17	**STYLED-103** - Discarding of all the changes ITERATION BACKLOG	Discarding of all the changes	Low	FROZEN	2
19	**STYLED-104** - Undo & redo DONE	Undo & redo	Low	FROZEN	3
24	**STYLED-94** - Editing of the variables DONE	Editing of the variables	High	FROZEN	3
25	**STYLED-95** - Preview for variables DONE	Preview for variables	High	DONE	5
27	**STYLED-97** - Preview for basic markup DONE	Preview for basic markup	High	DONE	8
Total:					**34**

Test Coverage

File	% statements	% branch	% functions	% lines
actions	100	100	100	100
App.js	100	100	100	100
ComponentEditor.js	50	100	0	50
ComponentPreview.js	50	100	0	50
ComponentsContent.js	50	100	0	50
ComponentsList.js	11.11	0	0	11.11
Footer.js	100	100	100	100
Header.js	100	100	100	100
Sider.js	100	100	100	100
ThemeContent.js	25	23.08	40	26.32
ThemesList.js	59.09	78.95	55.56	60
ComponentsContainer.js	50	100	0	50
ContentContainer.js	50	100	33.33	44.44
SidebarComponents.js	42.86	100	0	50
ThemesContainer.js	65.5	100	50	57.14
themes.js	100	85.71	100	100
initialState.js	100	100	100	100
themeState.js	100	100	100	100
Total:	**57.63**	**44.64**	**40**	**56.88**

Code Quality

Lines of Code: 1,112

Reliability rating: A
Security rating: A
Maintainability: A

Duplications: 18.3%

Technical Debt: 6 hours 3 minutes
Code Smell: 44 blocks
Bugs: 0
Vulnerabilities: 0

Iteration #5

Issues

Number	JIRA issue	Summary	Priority	Status	Story Points
17	**STYLED-103** - Discarding of all the changes ITERATION BACKLOG	Discarding of all the changes	Low	IN PROGRESS	2
19	**STYLED-104** - Undo & redo DONE	Undo & redo	Low	IN PROGRESS	3
24	**STYLED-94** - Editing of the variables DONE	Editing of the variables	High	IN PROGRESS	3
25	**STYLED-95** - Preview for variables DONE	Preview for variables	High	FROZEN	5
5	**STYLED-90** - Grouping of component's attributes ITERATION BACKLOG	Grouped attributes	High	IN PROGRESS	2
14	**STYLED-118** - Preview of the theme DONE	Theme preview	Could	FROZEN	13
Total:					**28**

Test Coverage

File	% statements	% branch	% functions	% lines
actions	100	100	100	100
App.js	100	100	100	100
ComponentEditor.js	50	100	0	50
ComponentPreview.js	50	100	0	50
ComponentsContent.js	50	100	0	50
ComponentsList.js	88.89	50	87.5	88.89
Footer.js	100	100	100	100
Header.js	100	100	100	100
Sider.js	100	100	100	100
ThemeContent.js	25	23.08	40	26.32
ThemesList.js	59.09	78.95	55.56	60
ComponentContentContainer.js	50	100	0	50
SiderComponentsContainer.js	71.43	100	50	66.67
SiderThemesContainer.js	72.73	50	60	70
ThemeContentContainer.js	50	100	33.33	44.44
themes.js	100	85.71	100	100

initialState.js	100	100	100	100
themeState.js	100	100	100	100
Total:	**65.81**	**58.62**	**59.18**	**64.81**

Code Quality (SonarQube)

Lines of Code: 1,926

Reliability rating: A
Security rating: A
Maintainability: A

Duplications: 14.3%
Technical Debt: 3 hours 24 minutes
Code Smell: 56 blocks
Bugs: 0
Vulnerabilities: 0

Iteration #6

Issues

Number	JIRA issue	Summary	Priority	Status	Story Points
3	STYLED-89 - Tree of components DONE	see a tree with all available components in the style editor	Must	DONE	5
11	STYLED-102 - Delete a theme DONE	delete theme	Should	DONE	3
14	STYLED-118 - Preview of the theme DONE	see a preview of the theme	Could	IN PROGRESS	13
24	STYLED-94 - Editing of the variables DONE	customise values of Ant Design variables	Must	DONE	3
25	STYLED-95 - Preview for variables DONE	see changes for the variables, that were applied to the book in preview	Must	IN PROGRESS	5
Total:					**29**

Iteration #7

Issues

Number	JIRA issue	Summary	Priority	Status	Story Points
14	STYLED-118 - Preview of the theme DONE	see a preview of the theme	Could	DONE	13
16	STYLED-119 - Pre-built themes DONE	have pre-built themes	Could	DONE	8
19	STYLED-104 - Undo & redo DONE	undo or redo my changes	Should	DONE	3
13	STYLED-117 - Exporting a theme DONE	export theme	Could	FROZEN	3

25	**STYLED-95** - Preview for variables DONE	see changes for the variables, that were applied to the book in preview	Must	DONE	5
Total:					**32**

Iteration #8

Issues

Number	JIRA Issue	Summary	Priority	Status	Story Points
12	**STYLED-116** - Importing a theme DONE	import theme	Could	DONE	8
13	**STYLED-117** - Exporting a theme DONE	export theme	Could	DONE	5
17	**STYLED-103** - Discarding of all the changes ITERATION BACKLOG	discard all changes	Should	FROZEN	3
26	**STYLED-96** - Editing of the basic markup styles DONE	customize properties of the text/block (basic markup)	Must	DONE	13
Total:					**29**

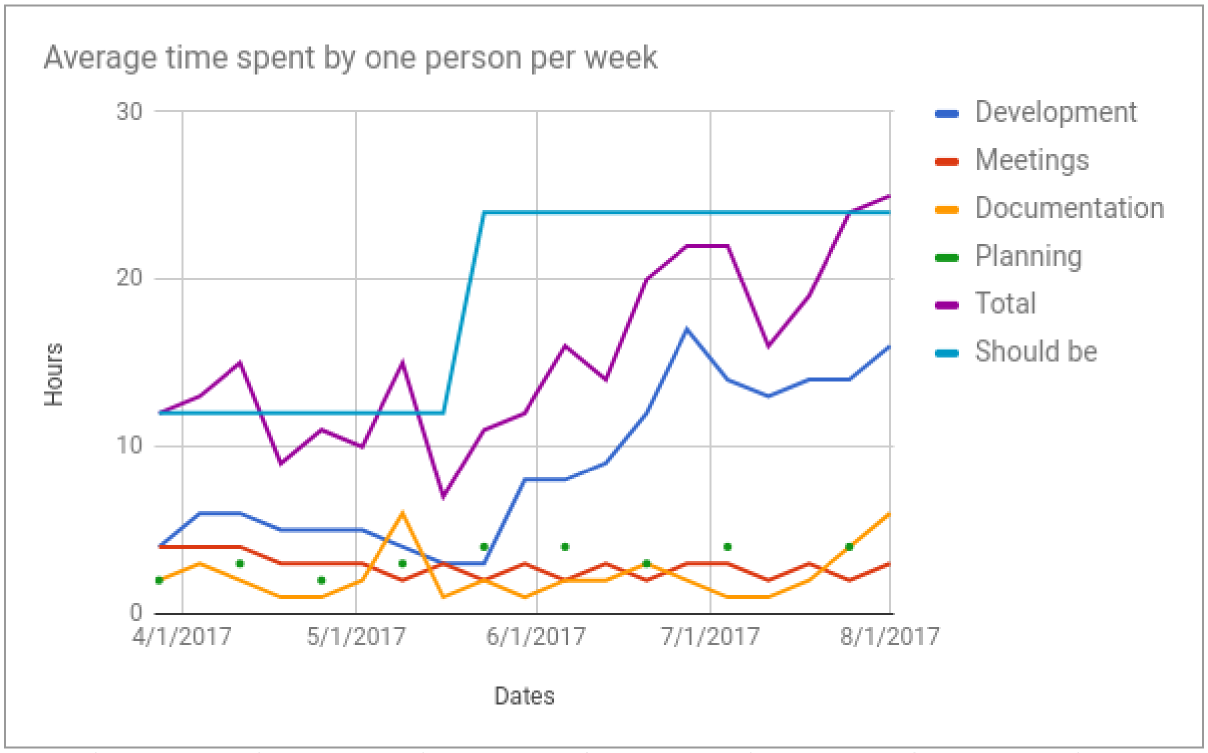
Iteration #9

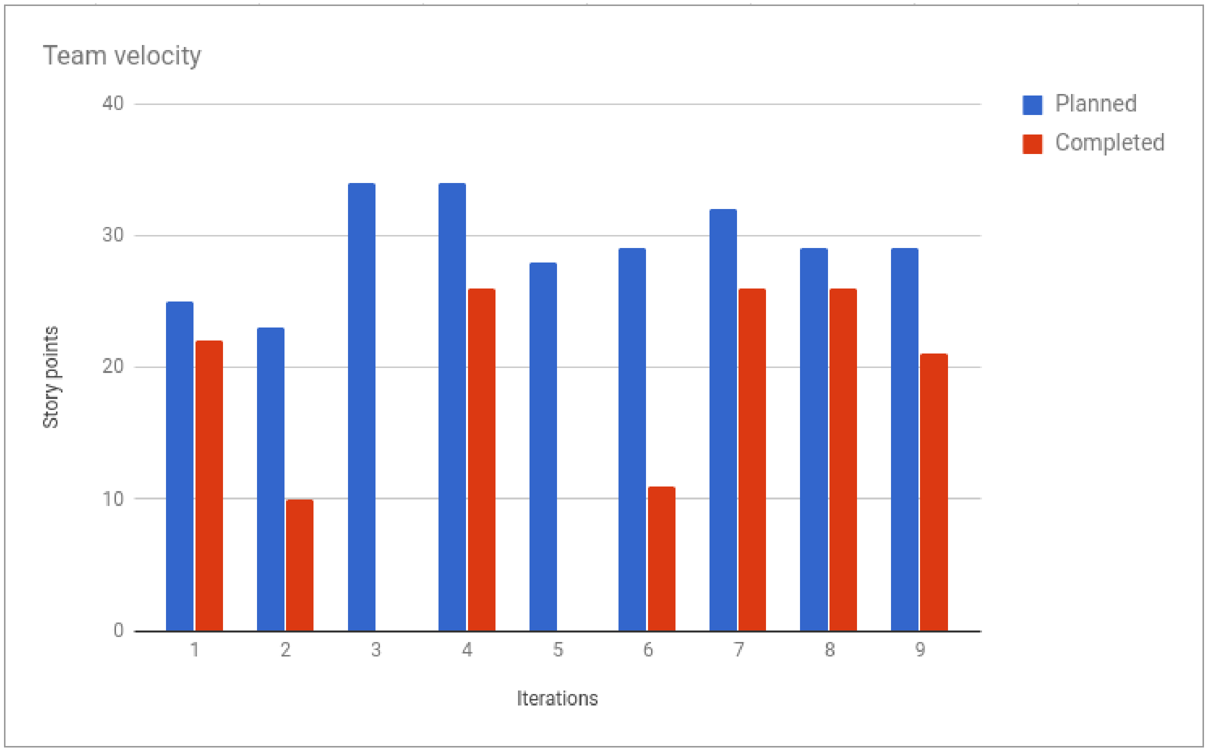
Issues

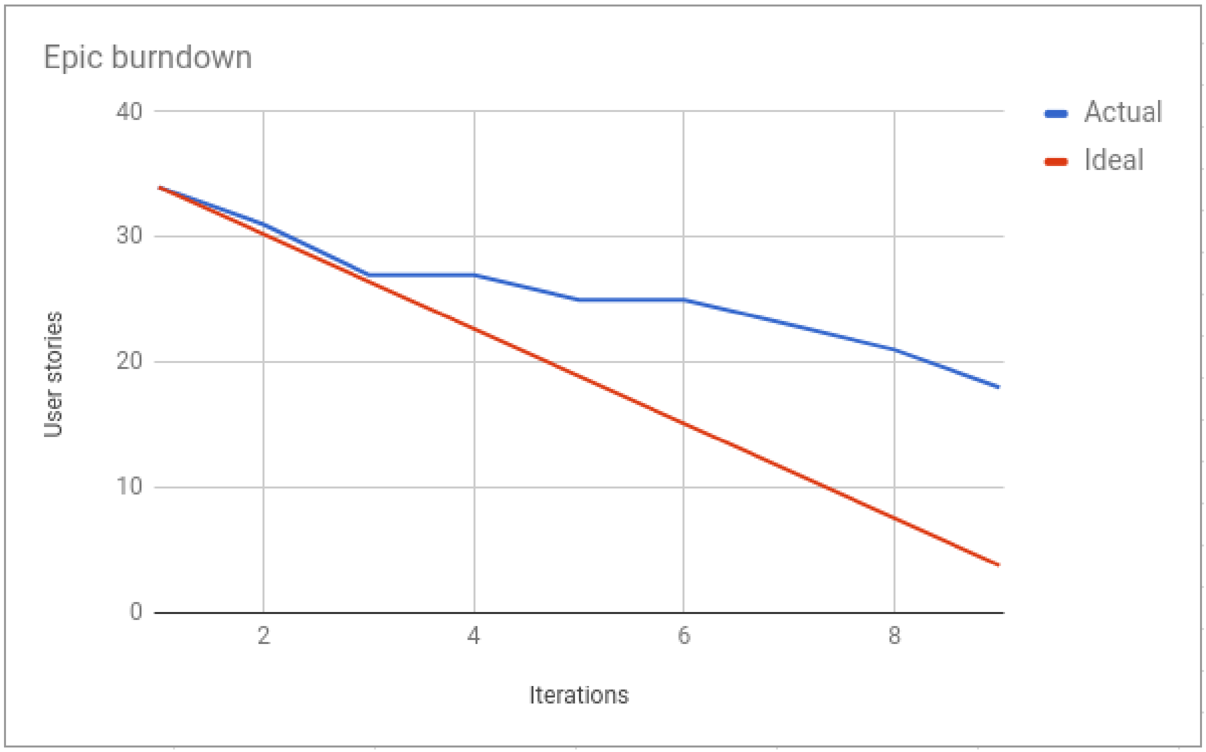
Number	JIRA Issue	Summary	Priority	Status	Story Points
17	**STYLED-103** - Discarding of all the changes ITERATION BACKLOG	discard all changes	Should	IN PROGRESS	3
20	**STYLED-93** - Theme applying DONE	apply selected theme to the textbook	Must	DONE	13
27	**STYLED-97** - PREVIEW FOR BASIC MARKUP DONE	see changes for the markup applied to the book in preview	Must	DONE	8
41		read help information	Should	IN PROGRESS	5
Total:					**29**

Metrics

Code metrics you can find in the Test report


Average time spent by one person per week
Hours
30
20
10
0
4/1/2017
5/1/2017
6/1/2017
7/1/2017
8/1/2017
Dates
Development
Meetings
Documentation
Planning
Total
Should be


Team velocity
Story points
40
30
20
10
0
1
2
3
4
5
6
7
8
9
Iterations
Planned
Completed



Architecture

Stack of technologies

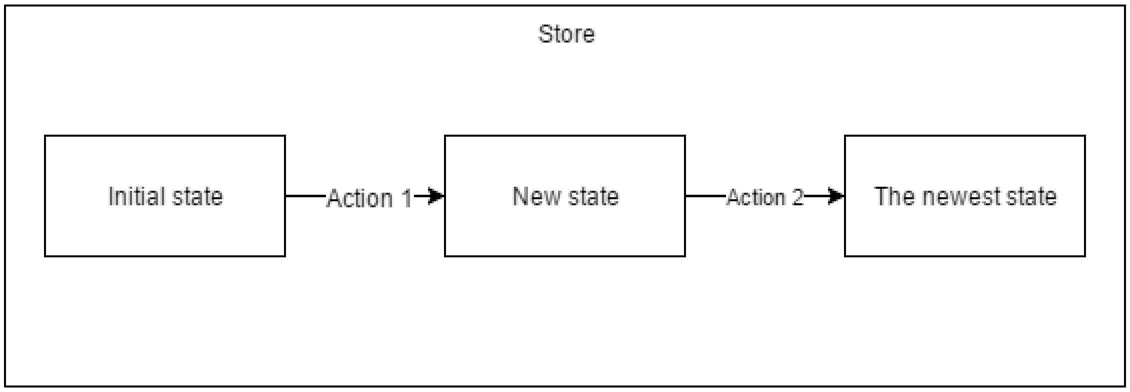
- React (view-library) [1]
- Redux (state container) [2]
- Ant.Design (UI framework) [3]

There is no need in database or backend, because all themes will store in the localstorage of a browser (or app). If user will need to move themes to another application he can use "Export" feature. Also, with only client-side it can work offline.

Used patterns

Redux approach

The main principle of Redux in the architecture of application, that we have the only one source of data - the store. All changes and operations are applied to this store and then application gets new state, which is based on the previous state and action, that was completed.



So, that approach allows the application to not have a mess of properties and values, store them as one JSON-object, and all action will be predictable.

Container pattern

Also, Redux allows us to use one more approach of distinguishing view from logic - presentational and container components.

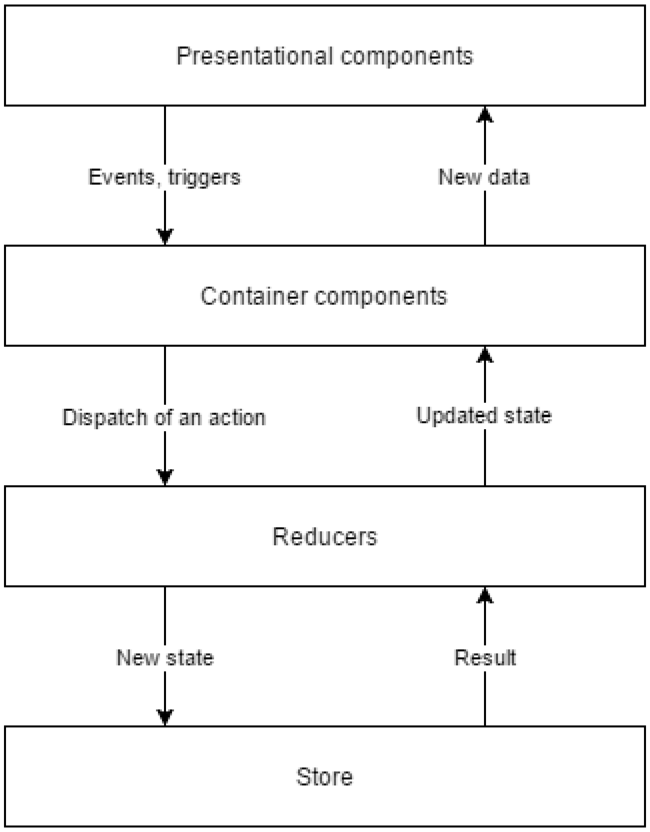
The main purpose of presentational components is how application looks like. Here you can find only JSX-code and data, which will be shown inside this JSX.

Container components get events from presentational components, call an specific action and use reducers for changing the state of the application accordingly. So, containers and reducers have all the logic of the application.

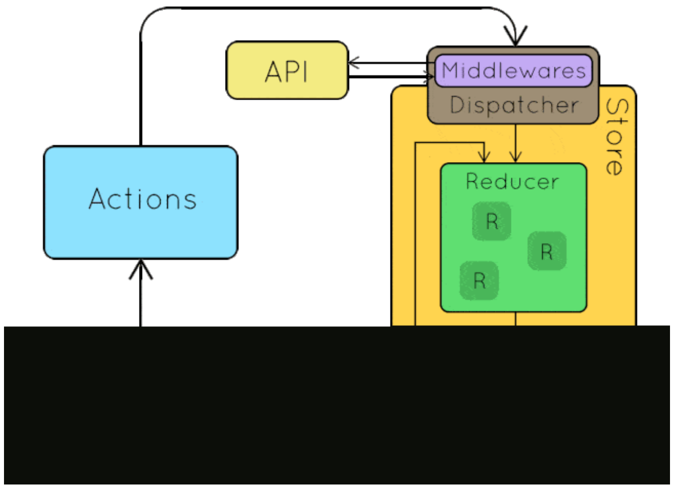
Almost MVC

In our case the architecture looks almost like MVC approach, but with some significant changes:

- View is presentationa; components
- Controller is container components. Application has more than one controller.
- Container components dispatch actions to reducers, so that reducers know, what operations they need to do with state.
- Model now just a store of all values. It doesn't work with changes, just gets the new one. When store is updated is sends new state to reducers.
- Container components subscribed on changes in store and when it's updated presentational components will get new attributed and will be rendered again.



Related documents



Reference:

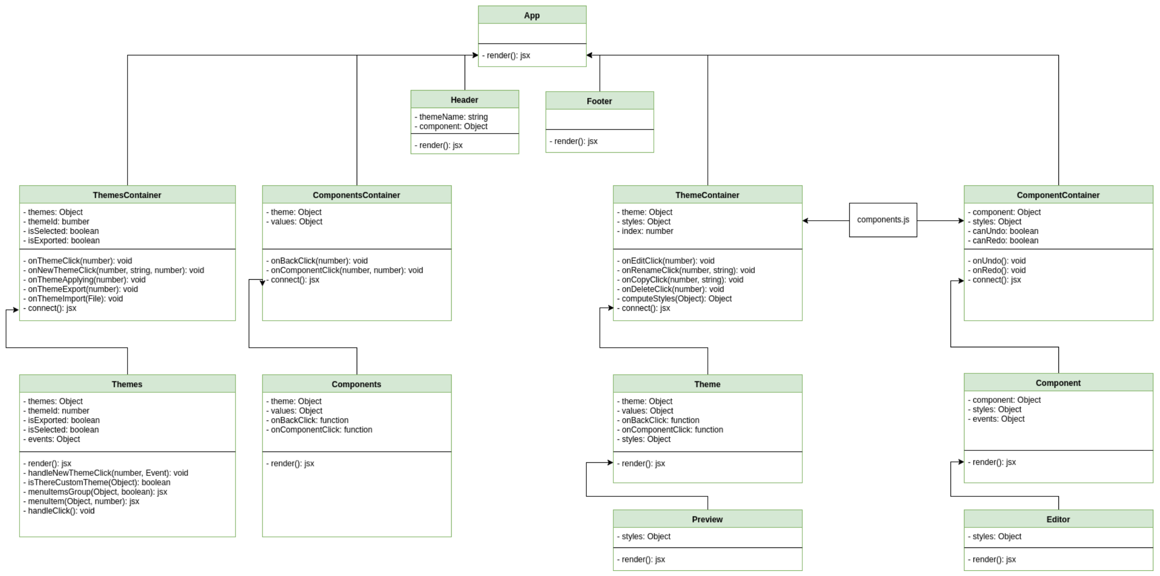
1. *Link:https://facebook.github.io/react/*
2. *Link: http://redux.js.org*
3. *Link: https://ant.design/docs/spec/introduce*

- Abowd, G.; Bass, L.; Clements, P.; Kazman, R.; Northrop, L.; & Zaremski, A. *Recommended Best Industrial Practice for Software Architecture Evaluation*(CMU/SEI-96-TR-025, ADA320786). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996.
- Bachmann, F.; Bass, L.; Chastek, G.; Donohoe, P.; & Peruzzi, F. *The Architecture Based Design Method* (CMU/SEI-2000-TR-001, ADA375851). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000.
- Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice, Second Edition*. Reading, MA: Addison-Wesley, 2003.
- Clements, P.; Kazman, R.; & Klein, M. *Evaluating Software Architectures: Methods and Case Studies*. Boston, MA: Addison-Wesley, 2001.

Static perspective

UML diagrams

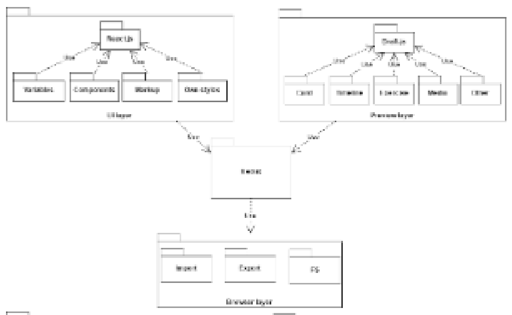
For checking other types of static perspectives use arrows on the top left corner of the diagram.

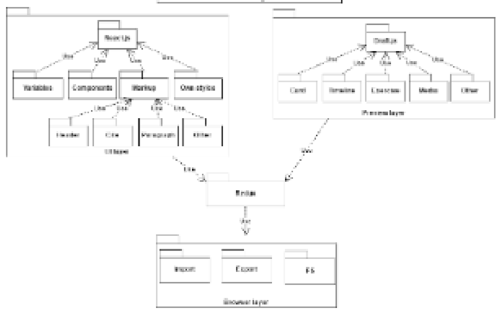


Description

The description of the static perspective

Previous versions



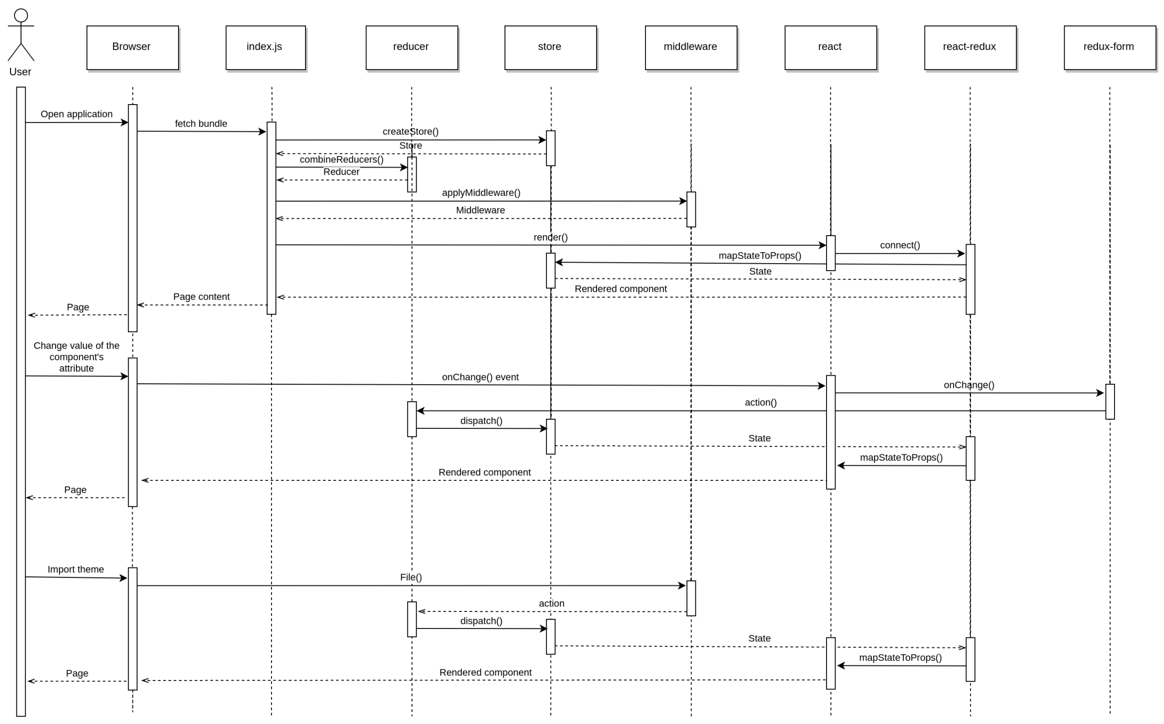


References

1. Documenting Software Architectures. Paul Clements, Felix Bachmann, Len Bass

Dynamic perspective

UML diagram



Description

On the sequence diagram you can see only three operation. Nonetheless, they cover approximately 90% of the functionality of the application.

1. First case when user open an application for the first time. It needs to create store for the application by combining all the reducers and apply middleware (for async operations and routing). After store is created it goes to react library via react-redux connector, which allows them to be notified when someone change state. In the end react returns rendered components, which can be used by browser.
2. Second case is changing attributes in the editor. When user click on an input, browser sends event onChange() to redux-form. It checks

the value of the input, make parsing and formatting inside itself and sends this value right to the reducer. Then reducer change state of the application and react sees through react-redux library, that values of the form has changed and load the recent one. Preview of the component sees those changes as well, because the store is a single source of true and all components get their parameters from it. All rendered components sends to the browser.

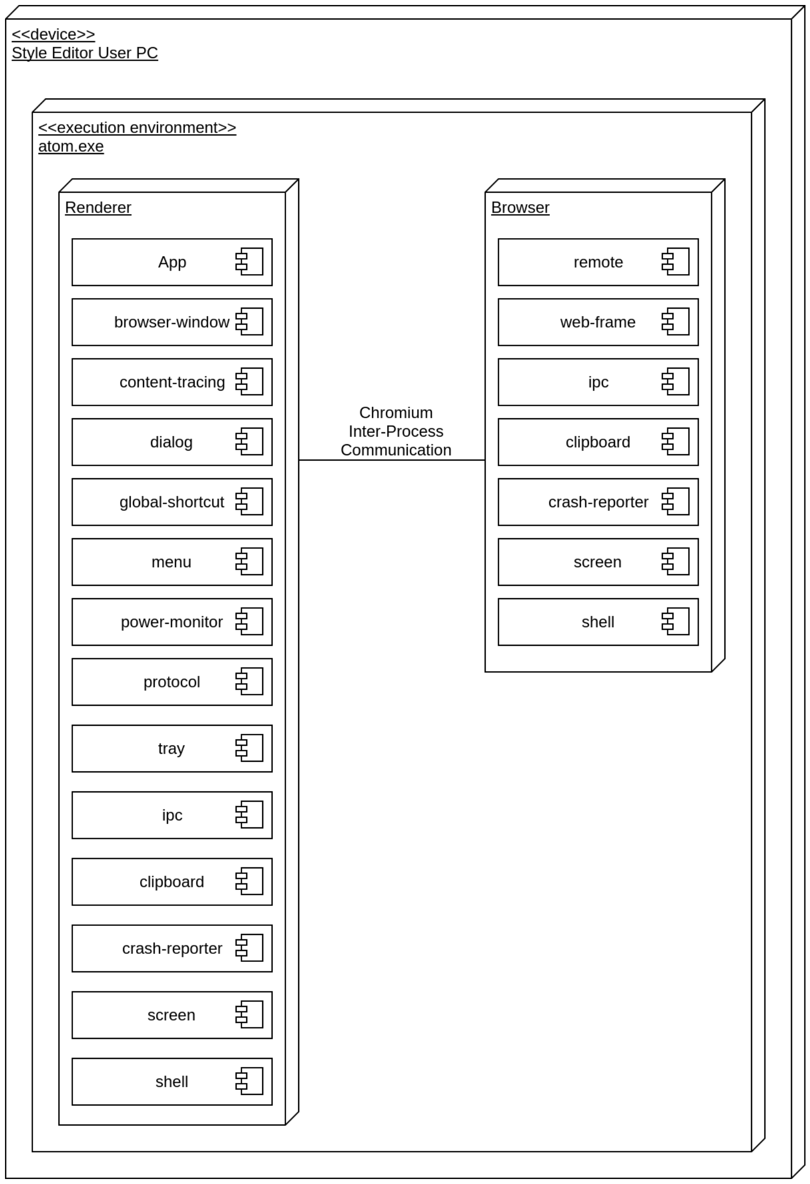
3. Third case when user wants to import a theme. Browser sees a file, that user wants to upload. We can't send a file's content right to the reducer, because it should be read - this is an async operation. So that we need to use middleware, which can work with async operation and only then dispatch an actions to the reducer. Then we get new state, which can be user for updating a page with the list of themes.

References

1. P. Clements, F. Bachmann, L. Bass. Documenting Software Architectures. Pearson Education, Inc., 2011.
   Link: http://ptgmedia.pearsoncmg.com/images/9780321552686/samplepages/0321552687.pdf

Allocation perspective

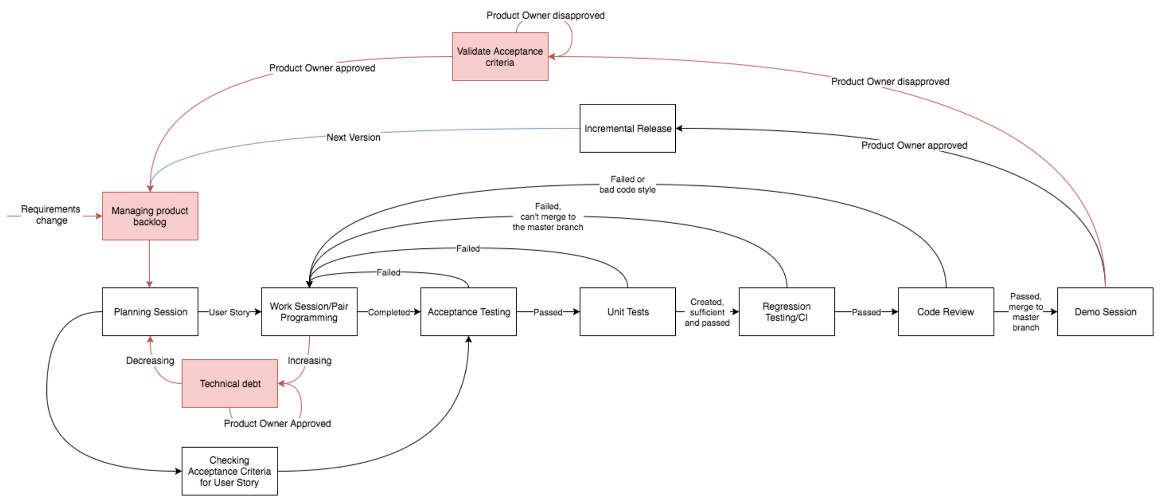
UML diagram



Description

In our case architecture of the application from the allocation perspective is limited by Electron framework and it's own architecture. Our part is an only one module in the whole architecture - App. Other modules are taken from the initial architecture of the Electron framework and it's bundled applications.

Quality management



`*Red colour - items added; Blue colour - items changed`

First of all, we have Work Item List with all the user stories, that were approved (even Acceptance Criteria for them) and prioritized by the Product Owner. Before any iteration, the team has Planning Session, where they discuss. which user stories will be made through this sprint, what should delivered in details, and planning poker is made. As a result, the team has a list of user stories for the iteration with priorities, acceptance criteria and estimations.

After the planning session team starts to work on selected user stories. Each team member can decide, will he or she starts to do it alone or will wait for the collective work sessions (time for them is planned beforehand) when team seats in one place for not more than four hours and code and in a case of difficulties any team member can ask for help. In a case of real troubles, a team member can ask for additional help through pair programming session.

Before the commiting developer should check, if done job satisfies Acceptance Criteria, because if it's not, then there is no reason to create unit tests, make code review and so on.

After completion of the user story unit tests should be created. Assignee for the user story should check list with functional requirements and tests for the selected user story. Based on that he should write new unit test of any predefined type.

After all new unit tests passed, branch with new functionality should be pushed to the new branch, so that Continuous Integration system will be able to check code style (syntax issues and so on) and do regressions testing. If it's failed, then code reviewer won't be able to merge this branch to the master branch, so that developer will see, that branch has some problems and they should be fixed before the code review.

If regression testing was passed, then it's time for code review, where another developer should check all conditions, code style, and logic. If everything is ok, then this branch will be merged into the master branch. If it's not, then problem parts of the code will be highlighted on the merge page in GitLab and assignee will be notified.

Technical debt should be managed during all development phase. It can be increased with Product Owner approval in purpose during the working session or it could be decreased through adding appropriate tasks during the planing session with Product Owner Approval.

The last step is a demo session when the team shows implemented features to the Product Owner. If he approved, then those user stories will be added to release branch, if not, then user stories and acceptance criteria should be verified, new acceptance criteria added or modified and changed for doing during next iterations.

Reference:

- J.M. Juran, A.B. Godfrey. Juran's Quality Handbook. Library of Congress Cataloging-in-Publication Data, 1998.
  Link: http://www.pqm-online.com/assets/files/lib/books/juran.pdf
- International Organization for Standardization & International Electrotechnical Commission. *Quality Management—Guidelines for Configuration Management* [ISO 10007:1995 (E)]. Geneva, Switzerland: International Organization for Standardization/ International Electrotechnical Commission, 1995.
- S. Thomas. Agile Quality Management, 2008.
  Link: http://itsadeliverything.com/agile-quality-management

Functional requirements

Due to our technology stack, we decided to use WebStorm IDE, that allows us to reduce unintentional bugs as typos, runtime errors, compile errors and suggest to create most critical exceptions. We also will use peer review to check team members code. This will help to learn the stack faster and find potential issues in the code.

Because, our team creates unit tests for every code unit, we have the ability to implement regression testing approach. Thus, due to setting up continues integration, all commits to the repository automatically checks for all unit tests that applicable for this piece of code

According to the Discipline agile, all artifacts evaluated by iterative approaches. Thus we focus on the first epic and clarify detailed user stories only for it. All functional requirements including acceptance criteria and and type of applicable test provided for each clarified user story respectively. We want to emphasise that, we do not provide acceptance criteria and test for *Epics* and *Features,* because they split to the user stories. (see pic.1)

Picture 1. Continuous integration

List of functional requirements

According to the fact that method which provided to all functional requirements is a Unit test, the testing environment is development and tool that use to perform that test is Jest (provided by Facebook), we do not include these columns in the table to avoid redundant content. We also assume that trending required learning how to use Jest tool.

#	Type	Description	Specific of the method	Tracking method
1	Epic	create a new style for e-textbooks		
1,1	Feature	have an intuitive interface in the style editor		
1.1.1	User story	be able to return to the editor of the textbook	White box testing	URL was changed, otherwise test failed.
1.1.2	User story	be able to return to the list of all the themes	White box testing	URL was changed, otherwise test failed.
1.1.3	User story	see a tree with all available components in the style editor	Positive testing	No errors occurs.
1.1.4	User story	distinguish default themes and own	Positive testing	No errors occurs.
1.1.5	User story	see all the edited attributes for each components grouped by the edited value	Positive testing	No errors occurs.
1.1.6	User story	change languages of the interface in the style editor	White-box testing	Put the possible (pre-determened) variables and check that text data was changed.
1.1.7	User story	see all available variables from the LESS-code	White-box testing	hecking all the variables in the LESS-file and that they are available from the model
1,2	Feature	work with different themes		
1.2.1	User story	see a list of all available themes	Positive testing	No errors occurs.
1.2.2	User story	rename theme	White-box testing	Put the different variables and check model that data was changed and theme was renamed.
1.2.3	User story	copy theme	White-box testing	Copy theme and check model that data was changed and theme was duplicated.
1.2.4	User story	delete theme	White-box testing	Delete theme and check model that data was changed and theme was mark as deleted.
1.2.5	User story	import theme	White-box testing	Import theme and check model that data was changed and new theme was added.
1.2.6	User story	export theme		Export theme and check it with sample.
1.2.7	User story	see a preview of the theme	Positive testing	No errors occurs.
1.2.8	User story	edit a theme	White-box testing	The URL was changed to style editor.
1.2.9	User story	have pre-built themes	Positive testing	No errors occurs.
1.2.10	User story	discard all changes		Discard changes and check that themes stay as it was.
1.2.11	User story	click on any component in preview in style editor mode and start to edit it	Positive testing	No errors occurs.
1.2.12	User story	undo or redo my changes	White-box testing	Apply undo actions and check if model can retrieve previous state. Apply redo actions and check if model can retrieve following state.
1.2.13	User story	apply selected theme to the textbook	Positive testing	No errors occurs.
1.2.14	User story	create a new theme	White-box testing	Check model that data was changed and default theme was duplicated.
1.2.15	User story	find a theme by the it's name	White-box testing	Search function works as it should

1.2.16	User story	see a spinner when changes are being saved	Positive testing	No errors occurs.
1,3	Feature	change variables of the framework		
1.3.1	User story	customize values of the variables	White-box testing	Checking, that variables are changed in the model if they were changed in the interface
1.3.2	User story	see changes applied to the book in preview	Positive testing	No errors occurs.
1,4	Feature	change styles of the markup		
1.4.1	User story	customize properties of the text/block	White-box testing	Checking, that variables are changed in the model if they were changed in the interface
1.4.2	User story	see changes applied to the book in preview	Positive testing	No errors occurs.
1,5	Feature	change the style of the components		
1.5.1	User story	customize properties of the components	White-box testing	Checking, that variables are changed in the model if they were changed in the interface
1.5.2	User story	see changes applied to the component in the preview	Positive testing	No errors occurs.
1,6	Feature	create own styles for the text for using in the editor mode		
1.6.1	User story	create own style	White-box testing	Checking, if new own style appeared in the model, if it was created in the interface
1.6.2	User story	rename the own style	White-box testing	Checking, if theme name was changed, if it was changed on the interface
1.6.3	User story	delete own style	White-box testing	Checking, if the removed theme is disappears from the model
1.6.4	User story	customize properties of the text/block	White-box testing	Checking, that variables are changed in the model if they were changed in the interface
1.6.5	User story	copy existed style and create the new one based on it	White-box testing	Checking, that created theme is equal to the predecessor theme
1.6.6	User story	see changes applied to the text/block in the preview	Positive testing	No errors occurs.
1,7	Feature	write LESS-code for the theme		
1.7.1	User story	see, is inputted LESS-code correct or not	White-box testing	Validation of the LESS-code
1.7.2	User story	see highlighted LESS-code	Positive testing	No errors occurs.

Reference:

- Cagan, M. & Wright, A. *Requirements for a Modern Software Configuration Management System*. Irvine, CA: Continuous Software Corporation, currently Telelogic, Inc., 1992.
- Davis, A. M. *Software Requirements: Analysis and Specification*. Englewood Cliffs, NJ: Prentice-Hall, 1990.
- Ecklund, Jr., E.; Delcambre, L.; & Freiling, M. "Change Cases: Use Cases That Identify Future Requirements," 342-358. *Conference Proceedings of the OOPSLA 96*. San Jose, CA, October 6-10, 1996. San Jose, CA: ACM Press, 1996.
- Sommerville, I. & Sawyer, P. *Requirements Engineering: A Good Practice Guide*. New York, NY: John Wiley & Sons, 1997.

Quality requirements

List of quality requirements

Sub-category	**Priority**	**Source**	**Stimulus**	**Environment**	**Artifact**	**Response**	**Response Measure**	**Description**	**Reasoning**
Category: **Performance**									
Time behavior	High	User	Periodic events (except CSS generation)	Runtime	Style Editor	Actions are performed	Average latency is not more than 1 second	The application should be able to response in average in one second for providing acceptable time behavior	Customer requirement

Resource utilization	Medium	User	Periodic events	Runtime	Style Editor	Actions are performed	Average memory utilization is not more than 1Gb, average CPU utilization is not more than 20% (Intel Core i7 6500U 2500 MGhz)	The application should be able to use not more that mentioned resources for providing acceptable resource utilization	Customer requirement
Capacity	**Not applicable**		**Reasoning** (In our case the only metric that describe capacity can be the number of user that work simultaneously with the editor, but that's our of the requirements)						
Category: **Compatibility**									
Co-existence	**Not applicable**		**Reasoning** (Co-existence of our application completely depends on the web-browser. We can't somehow adjust how much resources it will give to our application)						
Interoperability	High	User	Theme applying	Runtime	Textbook editor	CSS-file is generated	Correctness of CSS-file	When user wants to apply theme to the textbook, he needs to click on the button "Apply" when theme is selected, then CSS-file should be generated by the application, and sent to the textbook editor, which should take it and apply to the textbook	For checking of this QA we need to validate final CSS-file. If it's correct, then textbook editor will operate normally. It's not, then this QA is failed
Category: **Usability**									
Appropriateness recognizability	**Not applicable**		**Reasoning** (The application will be provided by the government, so, there is no need to describe, why this it should be used for the end-user. Also style editor is s required if the client works with textbooks, that were created by our application)						
Learnability	High	User	User uses application for the first time	Runtime	Style Editor	User applied edited theme to the textbook	User applied edited theme to the textbook after five minutes of experimentation	It's really important to show to the user how simple is theme editing and it's applying, so we need to measure, how much time is needed for the slight theme editing and it's applying	Customer requirement
Operability	High	User	User find specific component for editing	Runtime	Style Editor	User found specific component	User should find desired component within 20 seconds	The work of user is based on editing components of the textbook, so, the main part of the job is to find right component to edit	Customer requirement
User error protection	Medium	User	User edit component's attribute	Runtime	Style Editor	User edited desired attribute	System rejects all the undesirable input data	Our user is a power user. Even thought, system should reject all the attempts to enter incorrect data	Customer requirement
User interface aesthetics	High	User	General interaction	Runtime	Style Editor	User interacts with the system	User should be satisfied by using this application in general terms	User should be satisfied by interaction with the application and how it looks and feels	Customer requirement
Accessibility	Low	User with the widest range of characteristics and capabilities	General interaction	Runtime	Style Editor	User interacts with the system	User can interact with the system without any problems and additional tools	The application should be pleasant for the people with the widest range of characteristics and capabilities	Best practices
Category: **Reliability**									
Maturity	Medium	User	Periodic events	Runtime	Style Editor	Application is not crashed	Amount of crashes per 12 hours of work is less than one	Based on the customer requirement, crashes of the application is still possible, but the amount should be less then it was mentioned before	Customer requirement
Availability	**Not applicable**		**Reasoning** (In general, it will be stand-alone application, so it will be available to use whenever user wants)						
Fault tolerance	Medium	Component of Style Editor	Omission or bug	Runtime	Style Editor	Application is not crashed	Amount of crashes per 12 hours of work is less than one	Based on the customer requirement, crashes of the application is still possible, but the amount should be less then it was mentioned before	Customer requirement
Recoverability	High	User	Start of the application after crash	Runtime	Style Editor	Unsaved data is recovered	Not more that 2 last changes can be lost	If the application crashed, it should be able to restore unsaved data	Customer requirement
Category: **Security**									
Confidentiality	**Not applicable**		**Reasoning** (Application doesn't use any type of authentication)						
Integrity	Medium	User	Editing of default theme	Runtime	Style Editor	Theme is not changed	Default theme was not edited	In our application the only place where integrity can be applied is the editing of default themes. In any case, user shouldn't be able to edit them and system should prevent those changes.	Customer requirement
Non-repudiation	Low	User	Periodic events	Runtime	Style Editor	Actions chain	Each action (except the first one) should have a predecessor	In our application we use approach, in which all the actions, that are applied to the model, based on the previous state of the model and we store all the changes for undo & redo operations.	Best practices
Accountability	**Not applicable**		**Reasoning** (Our application doesn't support accounts, so there is no need in this quality attribute)						

Authenticity	**Not applicable**		**Reasoning** (Application allows anyone to edit themes in the application)						
Category: **Maintainability**									
Modularity	Medium	Developer	Wishes to apply changes to the component	Development time	Source code	Changes made and Unit Tested	Changes are done in four hours maximum	The components of the application should not be enormous and should be easily edited in acceptable time	Process requirement
Reusability	**Not applicable**		**Reasoning** (Our application can be reused only in textbook editor, but consider that both applications use the same stack of technologies, there won't be any problems with reusing, except some basic with integration)						
Analysability	Low	Team lead	Change request	Analysis time	Source code	Conclusion for the change request, based on the available documentation	Documentation covers more than 85% of source code	For analyzing of possibility of change, source code should be covered by the documentation	Best practices
Modifiability	High	Developer	Wishes to apply new action to the model	Development time	Source code	Changes made and Unit Tested	Changes are done in eight hours maximum	The actions, that applied to the model, should be easily managed by the developers in acceptable time	Process requirement
Testability	Medium	Quality Assurance Engineer	Testing of Code Unit	Testing time	Code Unit	Results captured	85% as a minimum level of code coverage by tests for the selected Unit	For appropriate quality of the application we need to support high level of code coverage	Best practices
Category: **Portability**									
Adaptability	**Not applicable**		**Reasoning** (If web application, then it depends on browser - if it works on the selected environment, then application will work as well. If Electron application, then it will work for almost of any system: Windows, Linux or macOS.)						
Installability	**Not applicable**		**Reasoning** (If web application, then it can't be installed. If Electron application, then it will be provided by third-party installer for all supported operating systems.)						
Replaceability	**Not applicable**		**Reasoning** (It's niche product, so it can't be replaced or replace other products.)						

Verification methods

Category	Sub-category	Method	Specifics of the methods	Testing environment	Tool	Training required	Tracking method
Performance	Time behavior	Analysis	Dynamic analysis	HP ProBook 440 G3 (Intel Core i7 6500U 2500 MGhz)	Gremlins.js	Gremlins.js documentation, JavaScript	Gremlins.js allows us to generated infinite amount of events that will be applied to our application, so that we will be able to get time measurements for each action and calculate the average value
	Resource utilization	Analysis	Dynamic analysis	HP ProBook 440 G3 (Intel Core i7 6500U 2500 MGhz)	Gremlins.js	Gremlins.js documentation, JavaScript	When Gremlin.js works, we can measure utilization of computer's resources. If it exceeds maximal values, then test is failed
	Capacity	**Not applicable**					
Compatibility	Co-existence	**Not applicable**					
	Interoperability	Testing	Random testing	Testing stage	Jest	Jest documentation, JavaScript	Jest will throw random values into the input form and try to generate CSS-file. If this file is not correct, than test is failed
Usability	Appropriateness recognizability	**Not applicable**					
	Learnability	Testing	SUS	Production environment	-	-	SUS will allow us to measure user satisfaction by the application

	Operability	Testing	SUS	Production environment	-	-	SUS will allow us to measure user satisfaction by the application
	User error protection	Testing	Black-box testing	Testing stage	Jest	Jest documentation, JavaScript	Jest will throw not acceptable and acceptable values to the input form and check, is model accept them or not (so, test is failed)
	User interface aesthetics	Testing	SUS	Production environment	-	-	SUS will allow us to measure user satisfaction by the application
	Accessibility	Testing	SUS	Production environment	-	-	SUS will allow us to measure user satisfaction by the application
Reliability	Maturity	Analysis	Dynamic analysis	Testing stage	Gremlins.js	Gremlins.js documentation, JavaScript	Gremlins.js allows us to test application under random set of actions on our virtual machine, and if case of crash we can get trace of the actions
	Availability	**Not applicable**					
	Fault tolerance	Analysis	Dynamic analysis	Testing stage	Gremlins.js	Gremlins.js documentation, JavaScript	Gremlins.js allows us to test application under random set of actions on our virtual machine, and if case of crash we can get trace of the actions
	Recoverability	Testing	White-box testing	Testing stage	Jest	Jest documentation, JavaScript	Jest will take examples of saved data and will put it into the model, after that it will compare model and saved data. If it's different, then test is failed
Security	Confidentiality	**Not applicable**					
	Integrity	Testing	White-box testing	Testing stage	Jest	Jest documentation, JavaScript	Jest will try to change default theme and after that it will check the model for any changes. If model was changed, then test is failed
	Non-repudiation	Testing	White-box testing	Testing stage	Jest	Jest documentation, JavaScript	Jest will pass test cases into the input form and will check the trace of the pred ecessors for each action. If some action (except the first one) doesn't have predecessor, then test is failed
	Accountability	**Not applicable**					
	Authenticity	**Not applicable**					
Maintainability	Modularity	Analysis	Static analysis	Development stage	SonarQube	SonarQube documentation	Complexity index should be less than 50
	Reusability	**Not applicable**					
	Analysability	Analysis	Static analysis	Development stage	JSDoc	JSDoc documentation	Coverage of files by documentation should be more than 85%

	Modifiability	Analysis	Static analysis	Development stage	SonarQube	SonarQube documentation	Maintainability index should be more or equal that B and Technical Debt should be less than 1 day
	Testability	Testing	Code coverage	Testing stage	SonarQube	SonarQube documentation	Coverage index should be more than 85%
Portability	Adaptability	**Not applicable**					
	Installability	**Not applicable**					
	Replaceability	**Not applicable**					

Previous versions

Quality requirements - v1.0.pdf

Reference:

- Dorfman, M. & Thayer, R. H. *Software Requirements Engineering*. Los Alamitos, CA: IEEE Computer Society Press, 1997.
- Sommerville, I. & Sawyer, P. *Requirements Engineering: A Good Practice Guide*. New York, NY: John Wiley & Sons, 1997.

Defect management process

Our defect management process include following steps:

- Defect prevention
- Baseline delivery
- Defect discovery
- Defect resolution
- Process improvement

Let's go through them one by one.

Defect prevention

First of all, it's important to understand, that it's easier and cheaper to prevent bugs, vulnerabilities and errors than fix them later. For this we use some tools and approaches.

After writing some peace of code we need to write appropriate test for that with amount of test cases. All the tests approaches were mentioned before. Then source code should be reviewed by ESLint, for maintaining common code style across the whole application. Then it can be commited and sent to the remote repository, where CI will run regression testing, then new test cases, then check code style, and, finally, will try to build application. In case of any error, this branch with commit will not be able to merge into the master branch. If all verification are passed, then it can be review by code reviewer (it should be another teammate). It code review is passed, then branch will be merged into the master branch.

Baseline delivery

If branch was merged into the master branch, then the product is baselined - it will be automatically deployed onto the development production for checking by customer. Now all bugs should be documented in the issue-tracker (in our case it's JIRA) and, depending on the priority, it should be resolved during current iteration (if bug is critical or high priority), or during next one (if it's from medium to lowest priority bug).

Defect discovery

In case of bugs in the implemented feature, it should be resolved within the task. If it appeared during regression testing, then it should be sent to the queue of bugs, as it was mentioned before.

Defects can be discovered not only by CI system, but also by our customer and by ourselves.

Case when end users can found bugs is not quite right, because our application won't be delivered to them because project will last until the end of the third semester, so we son't be able to got a feedback from them.

Defect resolution

As we mentioned before, bugs are divided onto several groups by priority (critical, high, medium, low, and lowest).

Critical bug	Description-1	Description-2	Examples
Critical	if it's breaks normal operation of the application of functional requirements with high priority (according to our backlog)	• Must be fixed immediatelly • Requires notification for PM • Daily follow up until closing	Application is getting started Critical business flow got stuck Data loss business loss loss of end user a security violation system is very unstable
High	if it's still possible to use application under normal operation in some cases or it breaks functional requirements with medium priority	• Serious • Must be included in next iteration • Requires notification for PM • Daily follow up until closing	Feature is missing which is not critical for only very few users that do not have much impact on business
Medium	if it's application works, but some results are unexpected	• Moderate • Must be scheduled two or three iterations	• Feature is missing but workaround is there • Feature of the functionality is not working
Low	if it doesn't influence on the normal operation somehow and user can use application for any kind of operations	• Low priority • Schedule when time is available	• It is impacting minor things, but, user is able to do the work
Lowest	if it's even can't be noticed by the user	• May never fix • Nice to have	• A cosmetic error • spelling mistake • typo • *Here, one important point to note is that even a spelling mistake or typo can also be of high priority or critical.

Based on the priority it will be sent to the queue of bugs resolving .

Process improvement

After the end of each iteration we check, why bugs are happened and how our process of defect discovery can be improved.

Reference:

- Grady, R. B. *Practical Software Metrics for Project Management and Process Improvement*. Englewood Cliffs, NJ: Prentice-Hall, 1992.
- Humphrey, W. "Justifying a Process Improvement Proposal." *news@SEI interactive 3*, 1 (March 2000).

Quality Assurance activities

Unit testing

For unit testing we use tool Jest, which is oriented on our stack of technologies and allows us to easily test React components.

According to our process, each developer should cover by tests any functionality, that was written and can be covered.

Code quality

For supporting appropriate code quality we use two tools: ESLint and SonarQube.

The first one allows us to follow common guidelines for code style and in case it's not corresponds with the selected one, it can be automatically fixed by this tool. For this tool we use JavaScript Standard Style, so that any developer won't be embarrassed by the bad quality code.

The second one allows us to measure code quality on the long distance, so that we can estimate how much time we need for fixing the problems and control technical debt in appropriate frames.

Continuous Integration

CI connect three previous activities into one. When developer commits any change to the repository, remote server with CI run all the found unit tests, then check code style, then tried to build the application, and in case of success publish (for master branch only), so that customer can check our job. If CI failed on any stage, than this commit can't be merged in the master branch and won't be staged for the production or deployment environment. This activity allows us to prevent many of both intentional and unintentional errors automatically.

Also, if some branch was merged into the master branch, then CI runs SonarQube scanner, so that we always have information about quality of the code in master branch.

Moreover, CI allows us to use regression testing, because CI get list of test suites from the repository and run them all for the whole application.

Pair programming & training sessions

Due to the lack of experience we have to increase our hard skills and we introduced to types of those activities:

- Training sessions - each iteration we gather fourth times when we do tasks and share our experience, ideas, and best practices. Any team member can ask for help from the team, so that everyone will try to dive into the problem and solve it.
- Pair programming is used in case of big and difficult tasks, which were not solved during training sessions. There are two main approaches of doing so: via Skype or face to face.

Peer Review

Before merge into the master branch, any commit should be approved by another teammate. Before commit, developer should create merge request in the repository for the task, then, after commit, it will be checked automatically by CI, and then it can be merged into the master branch by another developer, if everything from his point of view is ok. If it's not, then he can leave the comments to parts of code, that should be changed.

Acceptance Criterias & Demo sessions

When we create new functionality, we follow acceptance criterias, that mentioned in our backlog (that was approved by our customer) for each user story, so that we can estimate, is our implementation satisfies customer needs or not. In case feature is already implemented and it does not satisfy acceptance criteria, then it should redo.

Also, after each iteration we have demo with the customer, so that he can approve or not (if requirements were changed) implemented functionality.

User surveys

For testing some metrics for usability attribute we need to conduct user surveys. It will allow us to understand, is end user satisfied or not. Anyway, it can't be done right now, because we can't provide access to our application for the end users, because it still in development.

Documentation

For providing ease of understanding existing code we write documentation for each new component of the system. Documentation based on the JSDoc system and it can be easily converted into appropriate formats.

Reference:

- Beizer, B. *Software Testing Techniques*. Boston, MA: International Thompson Computer Press, 1990.
- McGregor, J. D. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001.
  Link: Testing a Software Product Line (CMU/SEI-2001-TR-022, ADA401736).

Required effort for QA

Type of requirement	Verification method	Quantity	Approximate spent time for the creating one test	Approximate spent time for all tests	Result (decision)
Functional requirements					
User stories	White box testing	19	30 minutes	10 hours	We decided to use this type of tests fully, because this requires for proper implementation of Continuous Integration. Write ones, this test runs every commit, so prevent us from coding mistakes. Thus, these types of testing save time for future debugging process.
	Black box testing	2	2 hours	4 hours	We decided to use this type of tests fully, because this requires for proper implementation of Continuous Integration. Write ones, this test runs every commit, so prevent us from coding mistakes. Thus, these types of testing save time for future debugging process.
	Acceptance criteria	Once, then only if requirements were changed	5 minutes	3 hours	We decided to use this type of tests fully, because this tests one of the requirements of our customer. This tests run ones during iteration delivery and provide feedback from customer how implemented functional corresponds to customer expectations.
	Demo presentation	After each iteration	1 hour	8 hours	We decided to use this type of tests fully, because this tests one of the requirements of our customer. This tests run ones during iteration delivery and provide feedback from customer how implemented functional corresponds to customer expectations.
Quality requirements					
Performance	Dynamic analysis	Writing once, doing each time before the delivery	5 hours	5 hours	Those verifications should be written once and then they can be used many times. They should be used during CI stage, so that won't require additional time from the developers and testers.
Compatibility	Random testing	Writing once, doing each time before the delivery	2 hour	2 hours	The main result of the work of our application is a CSS file and we can check the final result only by entering random data and then compare it with the final CSS, so, it should be done.
Usability	SUS	1	2 hours	2 hours	We won't be able to do SUS after the project because the project for our team will end after the development of the second epic. So that we won't use it.
	Black-box testing	Writing once, doing each time before the delivery	2 hours	2 hours	We will use this approach. It won't take too much time and can be done automatically each time after commit. Also. this quality requirements is important for the customer.
Reliability	White-box testing	1	30 minutes	30 minutes	In our case white-box testing is quite simple and doesn't take much time. Moreover, writing the white-box tests is a part of our process, so we will do that.
	Dynamic analysis	Writing once, doing each time before the delivery	5 hours	5 hours	This activity will be set up for the performance quality requirements as well, so there is no need in writing this test again.
Security	White-box testing	2	30 minutes	1 hour	In our case white-box testing is quite simple and doesn't take much time. Moreover, writing the white-box tests is a part of our process, so we will do that.

Maintainability	Static analysis	Setting up once, doing each time before the delivery	-	Automatically	Both of this types of testing can be done automatically by SonarQube tool on Continuous Integration server. We will use this tool in any case for checking other more important quality requirements and in the same time we can check those requirements as well.
	Code coverage	Setting up once, doing each time before the delivery	-	Automatically	

* Almost all estimations are based on the historical data after Iterations #0 and #1

Test report

Functional requirements

You may know that we gather test coverage automatically, so all numbers are calculated by the test tool - Jest. For some reasons it can't calculate correctly many functions, like dispatching actions to the reducer, browser-specific events, asynchronous operations, I/O operations, and many others, so the real number is bigger than in this table.

File	% statements	% branch	% functions	% lines
App.js	100%	59.21%	100%	100%
Footer.js	100%	75%	100%	100%
Header.js	100%	100%	100%	100%
HeaderContainer.js	100%	66.67%	100%	100%
Component.js	100%	100%	100%	100%
ComponentContainer.js	75%	100%	50%	75%
Editor.js	80%	100%	66.67%	80%
EditorContainer.js	85.71%	100%	66.67%	85.71%
Components.js	80%	100%	75%	80%
ComponentsContainer.js	62.5%	100%	50%	57.14%
Themes.js	46.43%	78.95%	38.46%	50%
ThemesContainer.js	57.14%	50%	37.5%	53.85%
Preview.js	100%	100%	100%	100%
Theme.js	23.81%	42.86%	40%	25%
ThemeContainer.js	80%	66.67%	42.86%	80%
themes.js	46.67%	39.13%	100%	46.67%
All files	**59.34%**	**59.21%**	**54.24%**	**60%**

Static analysis (SonarQube)

General statistics	*Code quality*	*Complexity*	*Quality attributes*

LOC: 5348 (JS), 119 (CSS) Statements: 280 Functions: 178 Files: 34 Commented: 3,3%	Bugs: 0 Vulnerabilities: 0 Technical debt: 0 Code smells: 0	Complexity/Function: 1.4 Complexity/File: 7.2 Cognitive complexity: 102	Maintainability: A Reliability: A Security: A

Quality requirements

Sub-category	Priority	Response Measure	Actual Response	Comments
Category: **Performance**				
Time behavior	High	Latency less than 1 second	~550ms	Checked on the most consuming operation - changing the attribute and rendering new preview
Resource utilization	Medium	Memory utilization less than 1Gb	Average is 130Mb	Checked with 1 themes in the state and full undo/redo (30 operations)
Category: **Compatibility**				
Interoperability	High	CSS-file is generated	CSS-file is generated	Generated CSS-file was tested with default and many custom themes on the example textbook. All changed values are successfully applies to the textbook.
Category: **Usability**				
Learnability	High	User applied edited theme to the textbook after five minutes of experimentation	-	We can't check it because we don't have access to users and data, that they can generate
Operability	High	User should find desired component within 20 seconds	-	We can't check it because we don't have access to users and data, that they can generate
User error protection	Medium	System rejects all the undesirable input data	If attribute is not correct, it won't change in the preview	During the project product owner decided to not implement checkers, but rather just ignore incorrect data
User interface aesthetics	High	User should be satisfied by using this application in general terms	-	We can't check it because we don't have access to users and data, that they can generate
Accessibility	High	User can interact with the system without any problems and additional tools	-	We can't check it because we don't have access to users and data, that they can generate
Category: **Reliability**				
Maturity	Medium	Amount of crashes per 12 hours of work is less than one	-	We can't check it because we don't have access to users and data, that they can generate
Fault tolerance	Medium	Amount of crashes per 12 hours of work is less than one	-	We can't check it because we don't have access to users and data, that they can generate
Recoverability	High	Unsaved data is recovered	Unsaved data is recovered	All changes in state are saved to the localStorage of the browser automatically, so that even if application would crash, it will restore all the data
Category: **Security**				
Integrity	Medium	Default theme was not edited	Default theme was not edited	Default themes can't be edited by the user - they are protected by redux
Non-repudiation	Low	Each action (except the first one) should have a predecessor	Each action (except the first one) should have a predecessor	Redux-approach are based on the actions, so we always can check the previous version of the state and which action was applied
Category: **Maintainability**				
Modularity	Medium	Changes are done is four hours maximum	Changes can be made in 10 minutes for any component	All components stored in one file and their properties are distinguished properly, so that they can be easily edited. Measurement was based on the team's data
Analysability	Low	Documentation covers more than 85% of source code	Documentation covers more than 95% of source code	We used JSDoc, so that documentation can be generated and bundled on the one page

Modifiability	High	Changes are done in eight hours maximum	Changes can be implemented in 2 hour	Based on the model of the application and redux-approach, changes are easily made by actions, that are dispatched to the reducer, which is responsible for changes on the state. Measurement was based on the team's data
Testability	Medium	85% as a minimum level of code coverage by tests for the selected Unit	Code coverage ~60%+	We can't measure exact value of code coverage. Check the first paragraph for reasons.

Configuration management

Branch management

It based on GitLab flow.

In general, we have those types of branches:

- **master branch** - main branch, where all the changes are merged
- **task branches** (naming "STYLED-number", e.g. STYLED-45) - those branches are used for doing tasks from backlog
- **release branches** (naming "major-minor-stable", e.g. 1-1-stable) - tasks for releases are based on backlog for versions from JIRA. At first, you need to finish all the tasks for the release and merge them into master, then create branch for the release. Release branch is created as late as possible and used for presenting the demo to the customer
- **production branch** - if you need to deploy your application, you need to push your changes into this branch from master

Path of every improvement will look like this:

1. Create a branch for the task, name is based on the name of task in JIRA
2. After the end of work with task create merge request into the master branch and choose someone, who should review your job
3. After the end of work with all tasks for the release, create new release branch and merge master into it
4. In case of hot fixes, create new branch as for ordinary tasks and merge it into the master or just use the same branch for dependent task
5. When release is done and you need to deploy it, merge it into production branch

Code Style

Based on JavaScript Standard Style

Possible improvements

- JSDoc
- Different environments for CI (development, production and test environments)

Change management

1. Change request should be gotten from the stakeholder in appropriate way
2. Change should be discussed inside the team for checking, how it will influence on the scope, stack of technologies, planning, and complexity
3. If all of these factors are not expensive, so that it should be implemented for the customer satisfaction
4. If it's too expensive, decision should be created in the Decision log and all pros and cons should be identified:
   a. If customer is agreed with the disadvantages, then it should be implemented
   b. If he is not agreed, then it shouldn't be implemented.

Release management

Disciplined Agile approach is to create potentially deliverable product at least after each iteration. If more often then even better.

For our project we decided to use Continuous Integration, so that we are able to deliver when we want. So that we decided to stop on two deliverances:

- Iteration releases - after each iteration branch with stable version is created and it will be deployed on the root of the server, so that customer can evaluate it by himself during or after the demo sessions
- Master releases - after code review and merge the task into the master branch it will be deployed on the development server, so that development team can use it for their needs

System building

Our product is self-assembling because of using some technologies, as:

- Babel - transpile all the dependencies into the common JavaScript standard, so that they can be used together [1].
- Webpack - gather all the dependencies and optimize them for the external use [2].
- Node - dependencies manager and compiler [3].

After compiling, that can be done on the Continuous Integration server (as it works now), system has only two files: .html and .js - that can be easily moved to any server.

Version management

It based on Semantic Versioning.

In brief, the main points:

- MAJOR version when you make incompatible API changes
- MINOR version when you add functionality in a backwards-compatible manner
- PATCH version when you make backwards-compatible bug fixes. Every time a bug-fix is included in a release branch the patch version is raised (to comply with Semantic Versioning) by setting a new tag

So that, each of the Epics consider, that we would have incompatible API changes in the final CSS file and exported file as well.

For the initial delivery of the first epic version naming is a following:

- 0.1 - UI and basic working environment
- 0.2 - work with themes
- 0.3 - work in editor
- 1.0 - export & import, LESS, CSS

Reference:

1. *Link: https://babeljs.io*
2. *Link: https://webpack.github.io*
3. *Link: https://nodejs.org/en/*

- Berczuk, Steve. *Software Configuration Management Patterns*. Boston, MA: Addison-Wesley, 2003.
- Dart, S. "Concepts in Configuration Management Systems," 1-18. *Proceedings of the Third International Conference on Software Configuration Managemen*t. Trondheim, Norway, June 12-14, 1991. New York, NY: Association for Computing Machinery Press, 1991.
- Institute of Electrical and Electronics Engineers. *IEEE Guide to Software Configuration Management* (IEEE Std 1042-1987). New York, NY: Institute of Electrical and Electronics Engineers, 1987.
- Ecklund, Jr., E.; Delcambre, L.; & Freiling, M. "Change Cases: Use Cases That Identify Future Requirements," 342-358. *Conference Proceedings of the OOPSLA 96*. San Jose, CA, October 6-10, 1996. San Jose, CA: ACM Press, 1996.
- Mockus, A. & Siy, H. "Measuring Domain Engineering Effects on Software Change Cost," 304-311. *Proceedings of Metrics 99: Sixth International Symposium on Software Metrics*. Boca Raton, FL, November 4-6, 1999. New York, NY: IEEE Computer Society Press, 1999.

- Ardis, M.; Dudak, P.; Dor, L.; Leu, W.; Nakatani, L.; Olsen, B.; & Pontrelli, P. "Domain Engineered Configuration Control," 479-494. *Softw are Product Lines: Proceedings of the First Software Product Line Conference* (SPLC1). Denver, Colorado, August 28-31, 2000. Boston, MA: Kluwer Academic Publishers, 2000.
- Release Management Best Practices.
  Link: http://www.plutora.com/blog/release-management-best-practices
- M.E. Bays. Software Release Methodology.
- J. Humble, D. Farley. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Addison-Wesley Signature Series (Fowler)).

- Brown, A. W.; Carney, D. J.; Morris, E. J.; Smith, D. B.; & Zarrella, P. F. *Principles of Case Tool Integration*. Oxford, U.K.: Oxford University Press, 1994.
- Software Engineering Institute. *Capability Maturity Model Integration, Version 1.1 CMMI for Systems Engineering and Software Engineering (CMMI-SE/SW, V1.1)*(CMU/SEI-2000-TR-018, ADA388775). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000.

- Berczuk, Steve. *Software Configuration Management Patterns*. Boston, MA: Addison-Wesley, 2003.
- Burrows, Clive & Wesley, Ian. *Ovum Evaluates: Configuration Management*. London, UK: Ovum, Ltd., 2005.

Change management

1. Change request should be gotten from the stakeholder in appropriate way
2. Change should be discussed inside the team for checking, how it will influence on the scope, stack of technologies, planning, and complexity
3. If all of these factors are not expensive, so that it should be implemented for the customer satisfaction
4. If it's too expensive, decision should be created in the Decision log and all pros and cons should be identified:
    a. If customer is agreed with the disadvantages, then it should be implemented
    b. If he is not agreed, then it shouldn't be implemented.

Reference:

- Ecklund, Jr., E.; Delcambre, L.; & Freiling, M. "Change Cases: Use Cases That Identify Future Requirements," 342-358. *Conference Proceedings of the OOPSLA 96*. San Jose, CA, October 6-10, 1996. San Jose, CA: ACM Press, 1996.
- Mockus, A. & Siy, H. "Measuring Domain Engineering Effects on Software Change Cost," 304-311. *Proceedings of Metrics 99: Sixth International Symposium on Software Metrics*. Boca Raton, FL, November 4-6, 1999. New York, NY: IEEE Computer Society Press, 1999.

Release management

Disciplined Agile approach is to create potentially deliverable product at least after each iteration. If more often then even better.

For our project we decided to use Continuous Integration, so that we are able to deliver when we want. So that we decided to stop on two deliverances:

- Iteration releases - after each iteration branch with stable version is created and it will be deployed on the root of the server, so that customer can evaluate it by himself during or after the demo sessions
- Master releases - after code review and merge the task into the master branch it will be deployed on the development server, so that development team can use it for their needs

Reference:

- Ardis, M.; Dudak, P.; Dor, L.; Leu, W.; Nakatani, L.; Olsen, B.; & Pontrelli, P. "Domain Engineered Configuration Control," 479-494. *Software Product Lines: Proceedings of the First Software Product Line Conference* (SPLC1). Denver, Colorado, August 28-31, 2000. Boston, MA: Kluwer Academic Publishers, 2000.
- Release Management Best Practices.
  Link: http://www.plutora.com/blog/release-management-best-practices
- M.E. Bays. Software Release Methodology.
- J. Humble, D. Farley. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Addison-Wesley Signature Series (Fowler)).

System building

Our product is self-assembling because of using some technologies, as:

- Babel - transpile all the dependencies into the common JavaScript standard, so that they can be used together [1].
- Webpack - gather all the dependencies and optimize them for the external use [2].
- Node - dependencies manager and compiler [3].

After compiling, that can be done on the Continuous Integration server (as it works now), system has only two files: .html and .js - that can be easily moved to any server.

Reference:

1. *Link: https://babeljs.io*
2. *Link: https://webpack.github.io*
3. *Link: https://nodejs.org/en/*

- Brown, A. W.; Carney, D. J.; Morris, E. J.; Smith, D. B.; & Zarrella, P. F. *Principles of Case Tool Integration*. Oxford, U.K.: Oxford University Press, 1994.
- Software Engineering Institute. *Capability Maturity Model Integration, Version 1.1 CMMI for Systems Engineering and Software Engineering (CMMI-SE/SW, V1.1)*(CMU/SEI-2000-TR-018, ADA388775). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000.

Version management

It based on Semantic Versioning.

In brief, the main points:

- MAJOR version when you make incompatible API changes
- MINOR version when you add functionality in a backwards-compatible manner
- PATCH version when you make backwards-compatible bug fixes. Every time a bug-fix is included in a release branch the patch version is raised (to comply with Semantic Versioning) by setting a new tag

So that, each of the Epics consider, that we would have incompatible API changes in the final CSS file and exported file as well.

For the initial delivery of the first epic version naming is a following:

- 0.1 - UI and basic working environment
- 0.2 - work with themes
- 0.3 - work in editor
- 1.0 - export & import, LESS, CSS

Reference:

- Berczuk, Steve. *Software Configuration Management Patterns*. Boston, MA: Addison-Wesley, 2003.
- Burrows, Clive & Wesley, Ian. *Ovum Evaluates: Configuration Management*. London, UK: Ovum, Ltd., 2005.

Reflection on feedback

1. Reflection on MOSP feedback

N	Focus	Score	Comments	Reflection
1	How much progress the team has made (25%)	**7**	1. The progress could be faster. The team just worked with customer requirements and changes. 2. Customer change requests should be managed more thoroughly. 3. Project progress should be measured systematically.	1. Show the progress of work, make accent on: • Coding part (sprints, US that done, burndown charts) • Quality (plan) 2. Emphasize on reached agreements with customer • Create document with our strategy • Create document with last changes (additional US that was added after PBL approved) • Sing or approve this documents by customer 3. Create a document: • with metric that we have to measure progress • additional metric that needed
2	Quality of the technical work (30%)	**7,5**	The quality is satisfactory. However, the quality plan could be more elaborate. More quality metrics should be defined for measuring the final project's result quality.	1. Provide QP that was made for Analysis, ask for assess and improvements. 2. Create a list of metrics that needed for measuring (according to SPM slides)
3	Software engineering discipline in terms of the practice areas (25%) (will be calculated as the average of the different practices)			
3.1	- project planning	8	Project planning is satisfactory. However improvement are needed in order to be more detailed. Project has to be managed at lower level of detail.	Create a list of metrics that needed for measuring (according to SPM slides).
3.2	- project tracking	6	Project tracking should improve significantly. The area has not been addressed sufficiently	Work thoroughly on this part, the most week of overall • Show progress in Confluence • Show progress in Jira • Investigate SPM slides for additional project tracking.
3.3	- requirements management and engineering	9	Very good	Keep same reflection strategy for EOSP.

3.4	- quality management	7	The quality plan should be: • more systematic • cover all aspects of quality.	• Same as for *Quality of the technical work.* • Investigate SPM slides for additional quality management.
3.5	- configuration management	7	More details on how the team is planning to work should be provided. Change management has not been addressed	Investigate SPM slides for change management.
3.6	- architecture design	7	The architecture should be details and better described.	TODO: add there perspective of the architecture.
3.7	- risk management	7	More specific risk reposnse is needed specially for high impact risks	• Systemize risks elicitation methods (add sections *Context*, *Risk Identification*, *Risk Analysis*, *Risk Evaluation*, *Risk Treatment*) • Update risks table • Add problems sections (risks that has materialized)
4	The team's reflection (10%)	**7**	Team has to reflect more on how handle changes and manage customer expectations.	Same as team progress, section 2.
5	Quality of the EOSP (10%) (actually this will be evaluated during the presentation)	**8**		Create presentation in advance, emphasize on the improvement part.

2. Reflection on EOSP feedback

N	Focus	Score	Comments	Reflection
1	How much progress the team has made (25%)	8.5	1. The team is on track. 2. Customer changes were managed satisfactory	Keep same reflection strategy as for MOSP, plus: • update documentation in time • for final presentation add summary and reflection
2	Quality of the technical work (30%)	8.5	The quality is satisfactory. New project metrics have been introduced in order to track the quality of the software.	Keep same reflection strategy as for MOSP: • use new metrics during second half of the project, provide the progress and result to customer and mentors
3	Software engineering discipline in terms of the practice areas (25%) (will be calculated as the average of the different practices)			
3.1	- project planning	8	Project planning is satisfactory.	Keep same reflection strategy as for MOSP, plus: • Finalise project planing • Add summary for the project planning • Decided what can be done better for future experience
3.2	- project tracking	8	Project tracking has improved significantly. • Product backlog is managed through JIRA, • project documentation is stored in Confluence New quality metrics have been introduced. However, these metrics have not been used yet.	Keep using JIRA and Confluence for project tracking. Use new metrics during second half of the project, provide the progress and result to customer and mentors
3.3	- requirements management and engineering	9	Very good. However, changes introduced by the customer should be managed systematically without disturbing the project practices and routines	In case that changes affect to the future progress significantly, revise the final acceptance criteria with customer, make changes in that document to reflect significant changes.

3.4	- quality management	8	A more elaborate quality plan was introduced. Evidence on how the introduced practices are applied are missing. Some minor issues are related to quality management: • I'd like to see top 5 QAs at most; • code coverage is about 60% which is not enough. Architecture: • dynamic view is missing • it's unclear whether they are going to add it	1. Show how practices in quality plan that we are introduced use in our project 2. Rearrange quality assurance based on the priority, and focus on the top five of them. 3. Keep code coverage on the 60%. 4. Add dynamic view to the architecture and keep focus on this perspective during development.
3.5	- configuration management	8	More details on how the team is/should be working have been provided. • Change management process has been introduced. • Evidence of applying the change management process is missing	Add to the confluence better explanations how the change management is used. (Put this information in chronological way, in separate blocks)
3.6	- architecture design	7	The architecture should be detailed and better described. No improvement after MOSP milestone.	Put significant effort to the architecture part. The Architecture folder was updated with Static perspective, Dynamic perspective and Allocation perspective pages, that fit current project implementation. Keep in mind this views during development phase and requirements negotiations with customers.
3.7	- risk management	8	Risk management is elaborate and risks are well defined. Evidence of risk monitoring and responses are missing.	Add to the confluence better explanations how the risk management and monitoring is used. (Put this information in chronological way, in separate blocks)
4	The team's reflection (10%)	**7**	Team handled changes and managed customer expectations rather satisfactory. However, it seems like all the members see the problems, but don't use the reflective practices much to solve them. However, they use feedback provided by mentors during the presentations and during the meetings.	Add reflection from MOSP to the confluence from internal documentation. Prepare same reflections for EOSP + add focus on the final presentation. (summary of the project, answer to the question "what can be done better for future experience?" Add information about meeting notes to the confluence and reflection to them.
5	Quality of the EOSP (10%) (actually this will be evaluated during the presentation)	**8**	The presentation is good. However, demo has failed.	Check demo in advance. Decide a variant to show video record instead of live presentation. (in sake of stability and time consuming)

3. Preparation for EOSP2

Type	N	Mentors suggestion	Reflection
Documentation	1.1	Create / update information in case of tracking and productivity	Update and finalise "Planning and estimation" document with all iterations in the Planning folder. Add last iterations in the iterations of the Tracking folder. Add summary and reflection to the Milestone plan_ with reflection part.
	1.2	Quality attributes (How we did and how our QA satisfy?)	
	1.3	Tests: • Testing and Acceptance testing • Produce test traces • Defects/errors amount • How many times doing regression testing?	Add a new page Test report in the Quality management folder to emphasise the procedure of test traces. Update the iterations of the Tracking folder with Defects/errors amount table.

	1.4	Reflections (what strategies was applied for mentors suggestions)	Added new folder "Reflection on feedback" where published strategies used on reflection for MOSP, added strategies for EOSP, EOPS2.
Presentation	2.1	Draft of the presentation in advance	Prepare draft of presentation and planned to show it to mentors 2 days before the presentation.
	2.3	• Show the project tracking info	We revised all iterations and finalise the "Planning and estimation" document, when shows the correlation between story points and hours, estimation of the project and analysis information. **Need to add:** most important information to the presentation
	2.4	• Show the problems (mistakes, well and wrong points): e.g. requirement changes etc.	Add summary of the lessons learned to the "Reflection on feedback".
	2.5	• Say about process: not present problems, but resu lts and lesson learned (what we had learned? designed? Planed in the beginning.)	Need to add: summary to the practice areas, to show obstacles that occurs during current practice area, how we deal with them and what to do in future.
	2.6	• Demonstrate the reflection, not just presentation the data, pass the message.	Add summary of the lessons learned to the "Reflection on feedback". Where focus on the most important aspects: • Resources • Management • Customer • Developing complementary software • Planing • Architecture • Documentation
	2.7	• What would have been done differently (If we'll do this project again?)	**Need to add:** the extra slide that will sum up lessons learned and project this information on this project.
	2.2	• Demonstrate the quality of work	**Need to create:** a video presentation with the demo of the product.
	2.8	• Speak about important features, which make the product attractive to customer.	Planing to show demo with most important features. **Need to create:** a list of them, consider in time maker and attractiveness.

4. Lessons learned

> "There is a process, but there is a real life too"
> © *Panos Fitsilis*

Resources

Resources one of the most important aspects that have to be considered before the project start and before the first iteration start.

In the case of the project:

The most problem was with software knowledge of people that have to work with the project. We manage it with reducing scope, how ever we cannot balance work in a way that everybody contributes equally.

In future experiences:

Before start project it's better to understand in which field the project will be created and answer the question: Do I have resources to work in this field?

Management

What ever type of SDLC you choose the management is the very important aspect. In agile every member of the team has to be highly motivated, self-organised, goal oriented. *However...*

In the case of the project:

In our project, every person has different priorities for the project, different goal and it was difficult to find levers that can affect people to work in most effective and efficient way. We do not pay a lot of attention to the each of us. Thus, why every of us feels lack of motivation in a different part of the project and the resulting decrease.

In future experiences:

We should not rely on self management concept that provided most of the agile methodologies and work hard to build a team, create motivation strategy and focus on team productivity as a manager.

Customer

According to agile, a customer should be highly involved in the project. *However...*

In the case of the project:

Customer not only highly involved but highly affect on the project as well. We make a mistake to work with two product owners at the same time. Thus, we have the different point of view on the same problem and spent a lot of time to find the final one.

In future experiences:

To eliminate such problems we need to separate stakeholder who make decisions and take up work and who might provide additional information or clarify some aspects that might help to elicit requirements.

Developing complementary software

In many cases, the product will be not the individual software but complementary one. That highly depends on the other software.

In the case of the project:

We have exactly the same case: to create a software that depends on the parent one. However, the parent software was not finalized. Thus, when the parent software changes, we have to change our part as well.

In future experiences:

Create a document with acceptance criteria and revise it with the customer, if one decided to change scope dramatically.

Planning

Most of the books recommend to create a plan and follow it. In agile, the plan is an iterative and incremental process that comes from general to specific. *However....*

In the case of the project:

Even general plans changes, and all that we think to do not only in development but organization way have big changes.

In future experiences:

Be ready that plan might and will change. Keep attention to the risk management, think about strategies that might be implemented. Revise all artifacts in a timely manner and use this artifact during the development process.

Architecture

From the very beginning the final architecture was not so clear: plenty of risks, communication issues, lack of stability from the customer side.

In the case of the project:

There are several problems with architecture of the application:

- Our part is the only one third of the ecosystem, that customer planned. Anyway, P2P didn't have any of them implemented. In the beginning of our project customer decided to change the whole architecture, so that we had to focus on the requirements. When we had to start the development, we still didn't have many important details about architecture.
- We could be sure that we would be able to implement any architecture, so we tried to avoid complicated things
- No one didn't know the whole stack of technologies, that was required by our customer
- There were some communication issues with lead-developer from P2P. When we tried to discuss some architecture issues, he was not interested and tried to offer the most obvious solution

In future experiences:

- Be familiar with stack of technologies: in case there are to many libraries and frameworks, we need to assign different teammates on learning different parts, so that we would always have at least one expert in library
- Map requirements and architecture in the beginning: check, how backlog corresponds architecture

- Start acceptance testing as soon as possible: for checking consistency of the final approach we need to start acceptance testing beforehand

Documentation

Documentation is an important process, according to agile philosophy all, there should be documentation that is needed for that project. The suggestion is tried to avoid essential documentation. *However....*

In the case of the project:

We ought to present standard artifacts, and this is good for the educational process, we need to know what documentation might be needed for different size and type of the project. But, it was challenge to explaining to the stockholders that we need to provide so much documentation for some small project (time that we spent to prepare documentation to subtract from the time that we can spend to code and test)

In future experiences:

We have to consider what documentation is needed and in which size. We have to follow the rule that we have to provide only essential documentation that will be used during the process and after it.
However, we knew know, that some artifacts might save a lot of time in the development if we consider to use them.