Applied Software Project Management

Project Plan

Team 2 members:

Noman Ashraf
Sai Priyatham Dongoor
Edgars Pelna
Pengyang Qi
Syed Mudassir Mansoor Shah
Shengjie Wang
Yuemin Zhao

Fall 2016

Table of contents

Note: all previous content of all sections has been completely rewritten.

1. Project introduction

1.1. Overview

- WowWeather is a desktop application for use on a home computer that shows the latest weather reports for any city around the world. It will include features such as daily or weekly weather reports, favorite locations, search history and more.

- The interface of the application will be user-friendly so that the end user can easily understand how to use it.

- The application will be programed in Java language and will fetch the weather reports from a publicly available API.

- The application will be free to use for everyone.

1.2. Motivation

The idea appeared when before one of the first meetings it started snowing outside. From this, the Team got the idea about a software for the latest weather reports.

In the Team, there are 7 members from which 6 members have Java background. Based on this, it was decided to develop the software in Java language.

1.3. Features

- Simple design
- Detailed 7-day forecast
- Works with any city in the world
- Location is shown on the map
- List of favourite locations
- List of recently searched cities

2. Software development methodology

The WowWeather will be developed using Agile sprints and different features will be delivered at different stages according to the schedule in section 4.1. Each member of the Team will be responsible for implementing different functionality.

The structure of project's management is directly affected by the deadlines and other deliverables of the WowWeather project. For this, Agile software development methodology has been chosen by the consensus of the Team.

2.1. Why Agile

We have selected Agile methodology for the WowWeather application because of it being more suitable for small size projects and allowing the change of requirements in the middle of the project. In our Team's case, the end result is not entirely clear in the beginning and an ability to do changes after the project has already started is what we need.

Since our Team has regular Status meetings and reports and investor is also in contact with the development team it feels natural to organise the project's lifecycle into easily adaptable iterations/sprints. Sprints allow us to modify our backlogs based on the feedback we get. Meetings are also more frequent during sprints and help discuss issues and mitigate risks that arise as soon as possible.

Moreover, team's skills are very different, and it seems a better idea to have sprints where at the end of each we come to a working prototype and see that it is functioning. Agile also means that the testing is performed alongside the development. This helps to produce a well-designed product in the end.

2.2. Methodology

There are different methodologies in Agile, and they have different advantages and disadvantages for different scope of projects.

Fdd	XP
Kanban	ASD
DSDM	Lean
Scrum	Crystal

Scrum is the most popular choice due to its simplicity with respect to other methodologies. It is proven that Scrum is productive and effective. In light of that, we have decided to use Scrum instead of any other.

For the project we have created "Product backlog" and prioritized the features in it. The sprint processes will start on 22nd of November and end on 7th of January which consist of 3 full sprints, each being 2 weeks long. At the end of every sprint period a significant part of the software will be completed and working.

During first meetings, the user stories have been divided into smaller tasks for which we performed the task estimation process with the help of Planning poker activity. Next meetings are held once every few days where all members of the Team can participate and join discussions about what has been done in the last 48 hours, whether problems have

appeared, and what will they do in next 48 hours. The data collected is used to draw the progress in the Burn down chart, further discussed in section 5.1. All members of the Team develop their features, then test them and after that commit to Github repository.

2.3. Delivery of WowWeather

In order for the Team to work together at the implementation stage, we have created a GitHub repository and every team member has been added as a collaborator. Each implemented task will be updated at GitHub repository so everybody can view and use features developed by others.

At the end of each sprint, all changes will be merged together to get a working demo with all the features that have been finished. This demo will be then shown to the investor.

3. Tasks

Our Team uses Agile methodology to complete the project as described in Section 2 of the document. That means that the Team will be maintaining two different backlogs - Product Backlog that contains all of the User stories for our product and will be managed by the Product Owner of the Team, and Sprint Backlog where some of the User stories from the Product Backlog will be put at the start of each sprint.

3.1. Product Backlog

Product Backlog holds only User stories without any specific tasks needed to perform in order to finish them. Meanwhile, for each User story in the Sprint Backlog there will be a number of tasks that our Team needs to perform as well as an estimated effort and the responsible person listed for each. In this document, our Team will also report progress for each task in the Sprint.

Table 3.1.1. is an extraction of User stories from the Product Backlog our Team has. Additional User stories might be added to it at later sprint planning meetings.

Table 3.1.1.

1. Code Backbone

1.1. As a team, we require a publicly available and free API for latest Weather Reports.
1.2. As a developer, I require a Network library so that I can send requests to the API.
1.3. As a developer, I need the response to be parsed for possible errors.
1.4. As a developer, I require the network response to be cached for some time to avoid request frequency limitations of the API.

2. Software Design

2.1. As a developer, I require a design sketch for Splash Screen (3.).
2.2. As a developer, I require a design sketch for Main Screen (4.).
2.3. As a developer, I require a design sketch for “About” section (8.) of the application.

3. Splash Screen

3.1. As a software owner, I want the user to be greeted by a welcome message before starting to use the main functionality.
3.2. As a software owner, I want the user to see the logo of the software on the Splash Screen.
3.3. As a software owner, I want the Splash Screen to be shown to the user only once so as not annoy the user.

4. Main Screen

4.1. As a user, I want the software to open in the middle of the screen so that I do not have to reposition it myself.
4.2. As a user, I want the window of the software to automatically be of size where the content fits in so that I do not have to change it myself all the time.
4.3. As a product owner, I want the window to have the design from User Story 2.2.

5. Weather Reports

5.1. As a user, I want to be able to search for locations that I want to see the weather report for.
5.2. As a user, I want to see the latest weather report for the current day for the location that I have found.
5.3. As a user, I want to see the latest weather report for a few days in the future for the location that I have found.
5.4. As a user, I want the weather report to contain some images, so that it is visually easier to comprehend the report.

6. List of Recent Searches

6.1. As a user, I want to see the list of my recent search history.
6.2. As a user, I want the latest weather report to be automatically shown for the most recent search when I open the application next time.

7. List of Favorites

7.1. As a user, I want to be able to add a location to the list of my favorite locations so that I can find it fast whenever I want again.
7.2. As a user, I want to be able to remove a favorited location from the list of my favorites.
7.3. As a product owner, I want the list to be of limited size, e.g., 15 locations max.

8. Extra

8.1. As a user, I want to be able to view the current version and the authors of the software.
8.2. As a user, I want to be able to choose the temperature scale in the settings, e.g., Celsius or Fahrenheit, so that the temperature in the weather report is shown in the selected scale.
8.3. As a user, I want to be able to choose the date format, e.g., AM/PM or 24 hours, so that the date and hours in the weather report are shown in the selected format.

IDs of the User stories used in table 3.1.1. are also referenced in Sprint Backlog shown in table 3.2.1. If any of those are changed in the Product Backlog, IDs will also be changed in the Sprint Backlog accordingly.

3.2. Sprint Backlog

For Sprint Backlog, a subset of User stories from Product Backlog will be chosen at the start of each sprint. Table 3.2.1. shows an example of tasks chosen for completing all the User stories under “Code Backbone” category.

Table 3.2.1.

US	Tasks	Responsible
1. Code Backbone		
1.1.	Task 1: find and compare weather report APIs	*Team*
	Task 2: select one API for our software	
1.2.	Task 1: create Network class	*Person x*
	Task 2: implement GET type of requests	
	Task 3: use API app id in each request	
	Task 4: return only JSON response to caller	
1.3.	Task 1: check for errors in network response and inform user	*Person x*
1.4.	Task 1: implement hashing for unique filenames	*Person x*
	Task 2: save network responses in cache files	
	Task 3: for each new request first look into cache	
	Task 4: use only recent files (e.g., 10 min old)	

The table does not list all the tasks for all User stories because tasks are written only at the start of sprints and many of them are not yet known at this time. Therefore, only an example of some tasks is shown. Also, the table excludes some of the details our Team includes in Sprint Backlog like effort estimation and responsibilities because those are described in more details in Section 4 and 5 of the document.

4. Effort estimation

This section describes the schedule the Team has chosen in order to complete the project till the selected deadline. It also shows the estimated effort needed to finish the tasks.

4.1. Schedule

As described in Section 2 of the document, for Agile methodology the Team will use sprints that are each 2 weeks long. Sprint schedule for the entire timeframe of the Project is shown in figure 4.1.1.

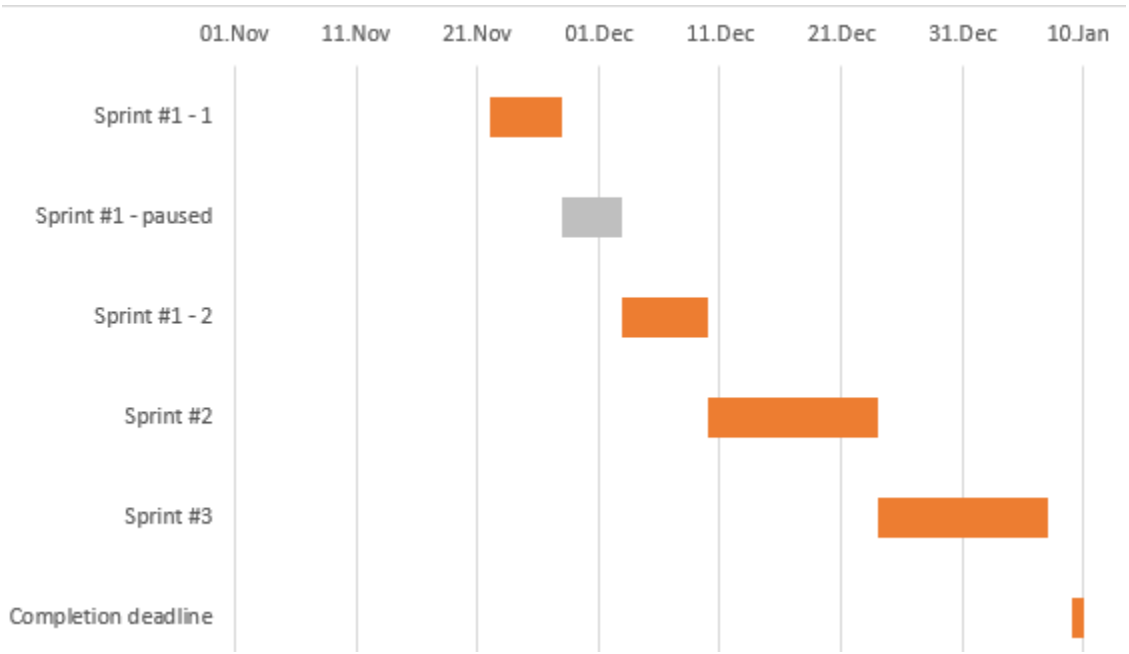

Figure 4.1.1.: Sprint execution schedule.

In the figure, there are 3 different sprints starting from the 22nd of November. The first sprint differs from the rest because there is a 5 days long pause in between. The sprint resumes on the 3rd of December and continues for another 7 days.

User stories in the Sprint backlog, as described in Section 3 of the document, reflect what will be implemented during any one of the sprints. When the next sprint starts, new User stories will be chosen from the Product Backlog.

After each sprint, when all of the sprint tasks are completed, the application will be in a working condition.

4.2. Estimated effort

The Team has already started the development in the first sprint and has selected some of the User stories from the Sprint backlog. For these stories, effort has been estimated in hours required to finish them and can be seen in table 4.2.1.

Effort is estimated during group's meetings where the tasks are discussed and evaluated. It is at this time when the Team tries to agree on the amount of hours the feature will require.

Table 4.2.1.

No.	User story category	Effort (in hours)
1.	Code Backbone	10.5
2.	Software Design	5
3.	Splash Screen	8
4.	Main Screen	4

Approximate effort has been estimated for other User stories in Product backlog as well but it is subject to change in the future sprint planning meetings when the Team will discuss required effort once again based on the experience from the previous sprints.

This effort as it currently stands can be seen in table 4.2.2.

Table 4.2.2.

No.	User story category	Effort (in hours)
5.	Weather Reports	20
6.	List of Recent Searches	7
7.	List of Favorites	7
8.	Extra	5

When all of the current Product Backlog's User stories will have been implemented, the Team will decide on new User stories that might make the software richer and estimate effort further.

5. Progress monitoring

If the project is going out of track then it leads to waste of time. To avoid this, our project is monitored and controlled keenly in the ways described further.

5.1. Burndown charts

This is the graphical representation for tracking the schedule and work already done. Planned schedule and actual on going schedule will be shown in the graph.

An example of a Burndown chart our Team will use is shown in figure 5.1.1. The chart is based on the length of Sprints discussed in section 4.1. The chart will be refreshed every second day to reflect the changes using the progress reported on Sprint Backlog as described in section 5.3.

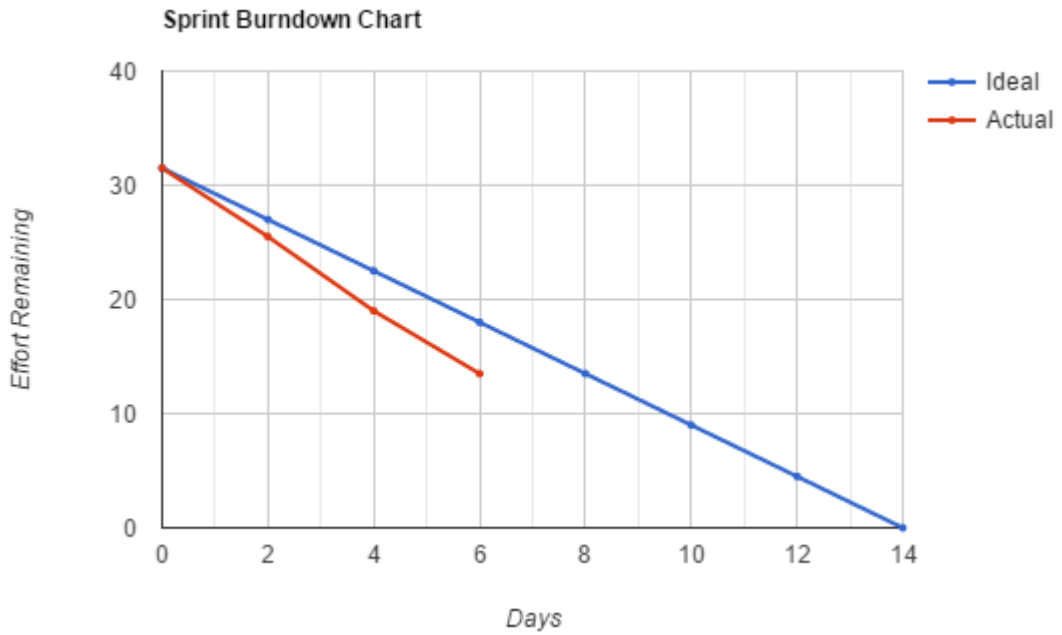

Figure 5.1.1.: example of a Burndown chart.

Comparing both lines shows if the development team is going in correct direction or is deviating from the project plan. A Burndown chart is to display the progress toward completion and give an estimate on the likelihood of timely completion.

5.2. Weekly status reports

Status report is sent to the product owner every week, which he will review and suggest developers if there is any requirement for changes. If the team is deviating from the project then the feedback from the status report helps them to review themselves and get into track.

5.3. Sprint Backlog table

To view progress status of each task a table is created which depicts our requirements, who is responsible for those particular tasks, progress of tasks every two days, and current position of the task - if it is started or ongoing, or completed. This helps us keep track of what has been completed, what should be developed next and how the work is going on.

An example of a Sprint Backlog progress monitoring can be seen in the figure 5.3.1.

US	Tasks	Responsible	Estimate	Day 2	Day 4
1. Code Backbone					
1.1.	Task 1: find and compare weather report APIs	Team	4	2	1
	Task 2: select one API for our software			0	0

Figure 5.3.1.: an example of progress monitoring in Sprint Backlog.

5.4. Other

Google Drive and GitHub repository also serve as a monitoring way, as the document uploaded on Google Drive and code shared through GitHub can be viewed by all the team members and suggestions or ideas regarding improving the project can be given by each and every team member.

Figure 5.4.1. shows how the Team monitors the progress of the completion of different documents.

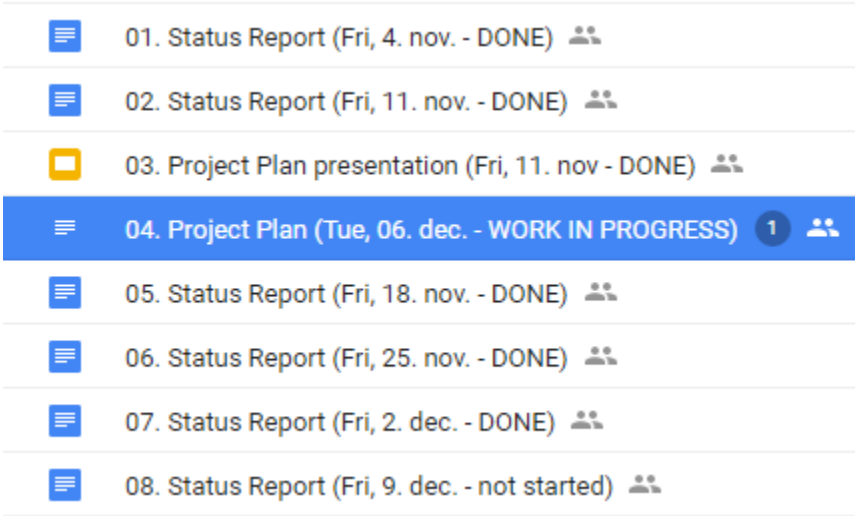

Figure 5.4.1.: example of how the Team monitors progress of document completion.

Apart from Scrum meetings, we have selected WhatsApp as the means of communication between team members so that their ideas can be shared instantly and knowledge can be transferred.

6. Risk analysis

This section of the document analyses the risks that may happen during the execution of WowWeather project as well as their mitigation strategies. The way how risk monitoring will be carried out is the following:

When:	During Sprint meetings. The Team will monitor and discuss if any of the risks has occured.
Who:	The Team together. If a risk occurs where a person cannot do something for some reason, it will be his/her responsibility to inform the rest of the Team as soon as possible.
How:	Team members will inform the rest of the Team of any issues that might have arisen doing their tasks. Team will also monitor how the rest of the team is participating.

The table 6.1. lists the risks that the Team has identified for WowWeather project and their mitigation strategy in the event they occur. For each risk, the likelihood of it occuring has been measured in group's meeting, choosing a value from the following scale:

- Very likely
- Likely
- May occur
- Unlikely
- Very unlikely

Table 6.1.

ID	Risk	Likelihood	Mitigation strategy
1	**Illness** Member of the group becomes ill.	Very likely	The person informs the Team about the situation right away. Depending of the seriousness, the person might still continue the development or participate in the meetings through conference calls. If work not possible, the Team replans the Sprint backlog.
2	**Being away** Member of the group is away and cannot participate.	Very likely	The person informs the Team about the situation as soon as possible. Situation is taken into account when planning Sprint backlog.

3	**Not participating** Member of the group is not actively participating in the project.	May occur	Team tries to find out the reason why the person is not participating. Reason and solution is discussed during sprint meetings.
4	**Missing meetings** Member of the group is not coming to meetings.	May occur	Team tries to find out the reason why the person is not coming to meetings. Meeting time can be changed, or person can join through conference calls.
5	**Meeting time** Team cannot agree on a meeting time when every member could join.	Likely	Team wil organize a few meetings a week at different times so that everyone can join at least every now and then.
6	**Incorrect effort estimation** During development, difficulties arise and the estimated effort turns out to be wrong.	Likely	Team discusses the situation during sprint meeting and possibly readjusts the Sprint backlog.
7	**Lack of experience** Group member lacks development experience, has only theoretical knowledge.	Very likely	Team member will be doing tasks that do not involve active development. If that is not possible, team members who have development experience will guide the inexperienced member and help him understand how to accomplish his tasks.

7. Quality assurance

In this section, a list of quality criteria has been listed and split into two parts - functional and non-functional criteria. Furthermore, it describes the activites that will be undertaken in order to ensure that the quality criteria are met.

7.1. Functional quality criteria

1. User stories from the Product Backlog are implemented by the set deadline.
2. The user can see the latest weather report for the location that he searches for.
3. The user can use the list of favorite locations.
4. The user can use the list of most recent searches.
5. Settings of the software are available to the user.

7.2. Non-functional quality criteria

1. On the day of the final presentation, the software is in a working condition.
2. *Efficiency* - the software limits the number of network requests it makes, caching response whenever possible.
3. *Usability* - from a selected number of customers, 80% of them can understand how to use the software in 5 minutes.
4. *Reliability* - the user is presented with a cached version of the weather report data when a new report cannot be downloaded.
5. *Fault tolerance* - in case of faults the software presents the user with an explanation of what went wrong and continues its work.

7.3. Quality assurance activities

1. *Regular meetings* - in these the Team estimates the effort required to complete sprints' tasks and reschedules the tasks if issues arise so that the core functionality is implemented by the selected deadline. In these, the Team also discusses the potential risks that might delay the successful completion of the software's tasks and whether they have occured.

2. *Code examination* - not all members of the Team are good at coding, therefore the Team will examine the code together during meetings to improve the quality of it.

3. *Testing* - this activity will be performed during each sprint so that at the end of the sprint the software is always in a working condition.

   The types of tests that will be performed include:
   - manual testing during development of the tasks;
   - manual testing when commits are merged from different branches;
   - regression testing, e.g., every next sprint the Team checks if the previous functionality still works;
   - unit tests will be executed for the main classes and functions of the software.

4. *User acceptance testing* - at the end the software will be shown to a selected number of customers that will try to use it. The usability will be measured according to the 7.2.3. criteria.