KTH- ID 2207 – Modern Methods in Software Engineering

Course Coordinator: Mihhail Matskin

Teacher Assistants: Shatha Jaradat & Lorenzo Corneo

Project Work - Fall 2015

SEP – Swedish Events Planners
USS Release Planning Document n.1

USS Developers - Group 16:

1. Alessio Russo, alessior@kth.se, 911103
2. Tobias Johansson, tobiajo@kth.se, 860901

Table of contents

Release planning №1

This is first release planning document describing the software system to be developed for SEP: Swedish Events Planner, a company in Sweden.

1 Vision

Produce a system able to implement the basic functionalities required to run the system, as requested by SEP.

The basic functionalities are:

1. Workflow of event requests (event initiation and application)
2. Client data management (client registration and events history)
3. Workflow of tasks distribution to services/production departments (choose one sub-team from each department)
4. Staff recruitment management.
5. Financial requests management.

The employees' records, scheduling issues, salaries and other parts of the problem are not required to be implemented for this release.

2 User stories

Bellow follows the extracted *user stories* for the project, according to the methodology of extreme programming (XP).

User Story Name	**Description**	**TE (GUI/ LOGIC)**	**Value/ Risk**
Login	As an employee of the company i can access the system functionalities through a login page where he/she enters his/her credentials. After verification, based on the account type, different functionalities are provided.	15m/30 m	H/L
New account	As part of the Management staff i can add new employee accounts to the system	15m/20 m	H/L
New request	As part of the customer staff i can create initiate new event planning request, by entering the event type, start and finish date and preferences about decorations, food, etc…	15m/20 m	H/L
View and search requests	As part of the customer service, administration team, financial manager, production and service manager can view and search requests	1h/40m	H/M
Employee reports	The vicepresident and the administration can make summary reports about employees' utilization	2h/1h	L/H
Client reports	The vicepresident,administration and financial manager can Generate reports about clients and events' statistics.	2h/1h	L/M
Organization report	The vicepresident can request reports from financial department about the organization situation	15m/30 m	L/H
Redirect requests	Customer Service Staff, Financial manager and administration can redirect requests according to SEP business description	15m/35 m	M/M
New client record	The customer service officer can create new client records, where the customer data like his/her name, etc… are saved into the system. This is called Client Request Details and it can be done only if a Client Request is already present into the system	20m/20 m	H/M

Search/View client records	The Customer service officer,Marketing team,Administration manager and financial manager can search and view for clients records	1h/40m	M/M
Receive/Update requests	The senior customer officer, financial manager, service/production manager and the administration manager can Receive/Update new event planning requests in the work items list.	30m/30 m	M/M
Events earning report	The financial manager and the administration can generate reports about event's earnings	40m/30 m	L/H
Search/View employee records	The HR team can search/view for employees' records	1h/40m	H/M
Make Discount	The financial manager can offer discounts to customers, it should be possible to do so with a web form from the user interface of the financial mananger, by selecting the customer and then set the new price	30m/30 m	L/H
Initiate task	The production/service manager can initiate a task for a sub team, requesting them to submit their plans regarding the event. THe manager can access this functionality from the request page. A priority can be assigned	1h/30m	H/M
Make Recruitment request	The production/service manager can request the HR team to hire or outsorce new people. This can be done from the interface of the manager	30m/20 m	M/L
Financial Request	The production/service manager can request the Finance team for more budget. This can be done from the request interface	30m/20 m	M/M
View Schedule	Managers can see the schedule of all the employee. This can be done from the main interface.	1h/1h	L/H
View tasks	Managers can see the tasks of a certain request after looking for the request. Production/Service employee can see the tasks assigned to them and the other tasks of that events.	1h/30m	M/M
Edit tasks	Production/Service managers can edit a task. Also a task can be edited by the employees working on it.	1h/30m	M/H
Add media content	Production/Service employee working on a certain task can upload media content to the database for that task/request	1h/1h	L/H

View media content	All employee, if not specified by the uploader of the content, can see all the media content in the database	1h/1h	M/M
Security	As an employee I want all my sensitive data to be encrypted so I can stay sure	Split	M/H
Edit account information	As an employee I want to be able to modify my account informations, like: my personal data, password, etc...	1h/20m	M/M
View discounts	The manager should be able to search for an history of the previous discounts offered to customers	1h/30m	L/H
Send Close task request	As an employee of the Service/Production manager I can send a request to close the task assigned to me to my supervisor	10m/20m	M/M
Close task	As a production/Service manager i can close a task after reviewing a send close task request	10m/15m	M/M
GUI	As an employee I want to use a GUI to perform all my actions	Split	M/H
Multiple operating systems	The system can be used on Windows, OSx and Linux systems.	Split	H/L
Synchronization	Whenever an employee updates a form that is being seen from another employee, the latter employee should be able to see the update in real time	1h/2h	L/H
Messages	The system should implement private messaging between employees	1h/1h	L/L
Manage recruitment requests	As part of the HR department i can manage recruitment requests submitted to me	30m/30m	H/L
Make RequestDetails	The customer senior officer can open new request details when a request is approved	30/30	L/H

Table 1. *User stories*

3 Planned release

For the first release an approximate of 20 stories where chosen out of the total 32 user stories in *Table 1*. The selection was based on the importance and on the risk, preferring low risk high value user stories in order to add as many functionalities as possible. A summary of the ammount of user stories in each category is given in *Table 2*. The selected stories are the following ones:

Login
New account
New request
New client record
Search/View requests
Search/View client records
GUI
Multiple operating systems

Redirect requests
Receive/Update requests
Initiate task
Recruitment request
Financial Request
Manage recruitment requests
Make Recruitment requests
Make RequestDetails

Make discount
View discounts
Edit tasks
Send Close task request
Close task
Edit account information

Value ***Risk***	**High**	**Medium**	**Low**
High	0	4	5
Medium	3	9	1
Low	9	1	1

Table 2. *Value/Risk summary of user stories*

4 Iteration planning

The choosen user stories was planned and implemented during three iterations, as seen bellow, with at least three refactoring in each iteration. The programming was planned to be done in *pair programming* there code is been reviewed continuously. Regarding testing was a test-driven approach (TDD) used during the project there automatic tests was written intially to intoducing logic functions to the system.

Iteration number - Name	1 - An affair of accounts and records
Iteration description	In this iteration the database necessary to hold all data is set up. All objects regarding Account, Client data and Request are defined. The database interface is defined and also the style of GUI.
Stories to be implemented	Login New account New request New client record Search/View requests Search/View client records GUI Multiple operating systems
Development time	3 days

Iteration number - Name	2 - An affair of requests
Iteration description	The rest of the diffrent types of requests are implemented and the user interfaces for handle them are definied.
Stories to be implemented	Redirect requests Receive/Update requests Recruitment request Financial Request Manage recruitment requests Make Recruitment requests Make RequestDetails
Development time	2 days

Iteration number - Name	3 - Marriage
Iteration description	The final implementations to deliver a useful protype.
Stories to be implemented	Make Discount View discounts Initiate tasks Edit tasks Send Close task request Close task Edit account information
Development time	1 days

5 Metaphore

The choose methaphore to describe the system is *Mailman*. The system is at its core to deliver requests and information between users of the system, like a mailman.

This metaphore was kept in mind when descions was taken of how to implemented interactions between the systems users.

6 Stand-up meeting report

Meeting Date	08 October 2015
Participants	Alessio Russo, Tobias Johansson
Meeting Notes	1. Summary of Our activities the previous day: a. Brainstorming around the project b. Studies of the project description and extreme programming c. Chose of programming language - Python 2. Today expected actions: a. Start a general list of user stories b. Start splitting and spiking the user stories c. Start to classificate user stories by value and risk 3. Problems: a. None of us has experience with GUI in Python for either desktop or web frameworks
Comments	

Meeting Date	10 October 2015
Participants	Alessio Russo, Tobias Johansson
Meeting Notes	1. Summary of Our activities the previous day: a. Finished the description of 30 user stories b. Investigation of solution for GUI implementation 2. Today expected actions: a. Add more user stories b. Start to classificate user stories by value and risks c. Brainstorm metaphoras d. Decision in GUI choice e. Continue work on report 3. Problems: a. Lack of time did not allow us to end the expected actions of the last meeting
Comments	

Meeting Date	12 October 2015
Participants	Alessio Russo, Tobias Johansson
Meeting Notes	1. Summary of Our activities the previous day: a. Learning tkinter (Python's de-facto standard GUI) b. Studies of project structure conventions with Python

	c. Improved a little the report 2. Today expected actions: a. Commit a project skelleton on GitHub b. Chose of storage solution for persistent data 3. Problems: a. One participant lacks experience with databases
Comments	The coding begins

Meeting Date	14 October 2015
Participants	Alessio Russo, Tobias Johansson
Meeting Notes	1. Summary of Our activities the previous day: a. MVC skeleton with basic basic GUI commited b. SQLite choosen 2. Today expected actions: a. Start with iteration 1 b. Find a system solution for access of persistent data for the View and Controller c. GUI for basic functionalites 3. Problems: a. No possibility to meet in person because of colliding schedules
Comments	

Meeting Date	16 October 2015
Participants	Alessio Russo, Tobias Johansson
Meeting Notes	1. Summary of Our activities the previous day: a. Solution found for access of persistent data: Database interface classes for persistent objects b. Database connection interface class 2. Today expected actions: a. Continue iteration 1 b. Database interface classes c. Fully functional login, new request and new client 3. Problems: a. No possibility to meet in person because of colliding schedules b. Iteration 1 takes much more than expected, mainly due to the database implementation
Comments	

Meeting Date	17 October 2015
Participants	Alessio Russo, Tobias Johansson
Meeting Notes	1. Summary of Our activities the previous day: a. Almost finished iteration1 b. Implemented Login, new client, new request 2. Today expected actions: a. Finish iteration 1, search request, search client b. Start iteration 2: Redirect requests, Receive/Update requests, Financial Request, Recruitment Request 3. Problems: a. How to implement search : for example, for client shoudl we use client id, email or name? best to go with email
Comments	Once the database started to be fully functional we worry only about the GUI

Meeting Date	18 October 2015
Participants	Alessio Russo, Tobias Johansson
Meeting Notes	1. Summary of Our activities the previous day: a. Finished iteration 1 b. Finished Financial, Recruitment request user stories. c. Finished Redirect requests, Receive/Update requests 2. Today expected actions: a. Continue and finish iteration 2 3. Problems a. Problems of communication on Skype, microphone problem
Comments	Thanks to the MVC architecture we can use the same view for multiple account, this helped us improving the development time

Meeting Date	19 October 2015
Participants	Alessio Russo, Tobias Johansson
Meeting Notes	1. Summary of Our activities the previous day: a. Iteration 2 completed 2. Today expected actions: a. Start iteration 3: tasks user stories b. Continue to work on the report 3. Problems:

	a. It takes very much time to implemented each type of request b. Alessio has a fever
Comments	

Meeting Date	20 October 2015
Participants	Alessio Russo, Tobias Johansson
Meeting Notes	4. Summary of Our activities the previous day: a. Implemented almost all the tasks user stories 5. Today expected actions: a. Finish iteration 3 b. Hand in report 6. Problems: a. (none)
Comments	

7 Tests

7.1 Model tests

All code was developed following the test driven approach. For every function that has nothing to do with the user interface (i.e. the model) there is a test. The tests are located in the folder test/model in the file ModelTests.py . All tests are run before the program is started in order to check if any bugs arises due to development changes.
Tests are done in a way that the output of a function is compared to the expected output of that function, using the “assert” command.

7.2 Acceptance tests

The chosen user stories to test are the following ones: Make RequestDetails and Make Recruitment Request.

7.2.1 Make RequestDetails Acceptance story

Input	**Output**
1. Open the software 2. Log in as sarah/sarah 3. Click on new client tab 4. Insert an email ‘test@gmail.com’ for a new client and click create and Fill in details for your client. Create 5. Click on view client tab, put the email of the client previously inserted and click get 6. Using the clientid shown in the View client tab,use that id in the new request tab. Click on new request tab, insert the client id an press Create 7. Fill in the details for the event and create 8. Log in as janet/janet(you have to open a new instance of the program) 9. In the first tab, Pending Requests, is shown a list of the pending requests. Search for the RequestID value 10. Click on view request and insert the request id and click get 11. Click on approve 12. Go back on pending requests and	1. The gui of the program opens, showing a username/password login form 2. The gui for the customer service operators appears 3. The form to create a new client account appears 4. All the forms appear and the data is saved (but no graphical output that the data is saved) 5. The client data appears, including the client id, which is 1 6. Forms to create a new request appear 7. Data is saved but there is no graphical output 8. Same as 1, but for the customer senior officer 9. A row should appear, where the first element is the RequestID, which should have value 1 10. The details of the request appear. A comment form appear 11. Data is saved 12. The pending request view should be empty 13. Same as 1, but for the financial

click update 13. Log in as alice/alice 14. In the first tab, Pending Requests, is shown a list of the pending requests. Search for the RequestID value 15. Click on view request and insert the request id and click get 16. Click on approve 17. Go back on pendingrequests and click update 18. Log in as mike/mike 19. Repeat same process from 14 to17 20. Log in as janet/janet 21. Repeate same process from 14to 17 22. Go on new request details tab and insert the requestID and click create 23. Go on view request details and put the requestID and click get	manager 14. should appear a row where the first value from the left is 1. 15. The detail about the request appear 16. The request is approved 17. Empty view 18. Same as 1, but for the administration manager 19. Same output of 14-17 20. GUI of the customer senior officer 21. Same output of 14-17 22. Forms for creating a request details should appeaer 23. The information of the request details should appear
Test result: All outputs as the expected result. Successful.	

7.2.2 Make Recruitment acceptance story	
Input	**Output**
1. Log in as jack/jack 2. Click on new recruitment 3. Fill in details and click request 4. Log in as simon/simon 5. Check pending recruitment tab and look for the recruitmentID value on the left 6. Click on view recruitment, insert the recruitment id that appaeared in pending recruitments and click get 7. Click on approve	1. The production manager gui appears 2. The form to create a new recruitment appears 3. The data is sent and the form closed 4. The HR gui appears 5. A row about a recruitment request should appear, with id 1 6. The details of the recruitment appear 7. The status of the recruitment changes to accepted
Test result: All outputs as the expected result. Successful.	

8 Conclusion and analysis

8.1 Pair programming process and refactoring

We had some difficulties first choosing the metaphore because of some doubts, in the end we have chosen a simple metaphora but because of the simplicity of the system it did not help us much.
Next we discussed how should be the software structured, if it should follow a MVC or Three layer architecture. We started with the idea of suing SQLITE as storage, so it would have make sense to use a Three layer architecture. In the end we used a mixture of the two, using MVC at the top layer of the Three layer architecture and use a DB interface to access the sqlite storage.
The stories in this document have the first estimation we have done, but after the first iteration it was clear that many of the estimated times could be halved, mainly because of the repetitiveness of the code.
Our pair programming process was a tough process, we could not meet many times and we mainly used Skype video conference and screen sharing to see the code in order to do pair programming. It is really helpful to use this kind of methodology, we could support each other and share ideas on how to improve the software, and by using Skype screen sharing we could also work during the evening. But it would been much easier of we sat in the same office 8-17 monday to friday and we lost some benefits because of that.
Another thing to mention is that Alessio had some background in Database Systems, whilst Tobias did not, so in order to speed up development we developed the system in a “sandwich” way, where Alessio developed the low level part of the system and Tobias the high level part. You would say that the hard part is to join the two things but actually it was very easy, because of the constant communication involved in pair programming. The refactoring mainly was applied to the DB interface in order to improve code in the beginning. After we used refactoring to solve GUI problems that we did not have thought in the beginning, and that sometimes involved changing the Database interface also.

8.2 Flaws in the system

The system has some flaws that were not addressed since it is the first release of the system:

1. No Input checking
2. No Encryption
3. Easy to reverse engineeer the code and inject code/read from memory

8.3 Comparison XP vs. OO analysis and design

This approach in our view is very good for prototyping and for developing system when short in time. We have some doubts whether this approach is good for large scale systems, but for a start point it is very good. Because of that many companies would not lose so much money when developing a system, since releases come sooner compared to the Waterfall model.

We can’t even talk much about one of the main point of XP: changing customer requirements. In this project requirements are not changing over time, so we can only do a “theoretical” comparison. From what we experienced in case of changes in the requirements it would be much faster to change the code if compared to the OO Analysis and design approach. It may require a complete refactoring of the system, but the more you code the system, the easier will be to refactor and avoid errors. In OO Analysis and design approach coding is one of the last steps, therefore in case of refactoring it would require so much time compared to the XP approach, since there is less insight into the system development with the second approach.

Appendix

1 Source code and libraries

Code found at: https://github.com/rssalessio/MMSE15Project-RussoJohansson

2 References

1. Lectures slides of the course "Modern Methods in Software Engineering" - 2015 Minhail Mitskin, KTH
2. SEP Business Documentation
3. http://www.extremeprogramming.org/