Individual Report for

Project IT 1003 Course

Evaluation of modern Software Development

Author: Zioris Konstantinos
zioris@kth.se

Information and Communication Technology
Stockholm 2017

Abstract

In this project a plugin for the Atlassian platform Confluence is created for calendar synchronisation. The work done in this project was based on the SGD developer framework. The individual work is assessed and evaluated based the four criteria, namely , the sum of effort spent on each individual activity within the whole project, distribution of the effort across activity types on an iteration basis, differences of effort spent on various activities in various iterations and comparison of distribution of effort spent on Non-Developer versus SGD Developer activities. The result showed that there three quarters of the time was spent on developer role activities while the non-developer activities the rest. First sprint was satisfactory despite the fact that the group had no prior experience in working according to SCRUM principles. The skeleton for the plugin was provided by the Confluence and the development language used was Java together with MySQL and Microsoft databases. Management of the project became more important as the end of the project came closer, while the collaboration with the client should be tighter and better for avoiding frustration and bad planning when the time run out.

Keywords: SCRUM, SGD developer framework, Agile software development

1. Introduction

Today's advances in technology, especially in the information technology have created the need for more complex developments which are only possible due to collaboration of bigger teams of individuals with different backgrounds and skills. For that to be feasible, various tools are created in order to create different ways/methods for facilitating and coordinating the work of the people. An example of that is the Atlassian Confluence platform. Confluence is a collaboration tool build for storing sharing and working on stuff. Today, someone has many documents to deal with, various notes for and from meetings, discussions and designs and schedules that are scattered pretty much everywhere. Confluence helps individuals, among others things, who work for software companies to more effectively organize these and in general their work.

Confluence provides tones of modularization possibilities through freely giving independent developers the possibility to develop plugins for adding functionality.

In this project a plugin in is to be constructed in order to achieve 2-Way synchronization between the Confluence calendar and the Microsoft Exchange Outlook calendar.

In the context of this project a 1-Way synchronization will render the basis for further development of a 2-Way connection in a future project. Toward this path an iterative agile development way of working , called SCRUM was used [1]. The goal of this individual report is to evaluate my personal experience through the most popular software development practices which I reported on a daily basis for all the SGD developer and non-developer activities. Pair programming, story driven working flow and time division into sprint periods are to be evaluated.

Chapter 2 presents the background and the goal of the project. Chapter 3 presents the methodology that was followed during the project development process. Chapter 4 the raw results while Chapter 5 includes some interpretation of them. Some improvements and future work concludes the chapter 5.

2. Background

In this project a development team consisting of 10 people, developed a plugin for connecting the Confluence build in calendar with the Microsoft Exchange Outlook calendar on behalf of the software company Betsson Group. The Confluence did not offer either 1-Way (Outlook transfer events to Confluence calendar) or 2-Way synchronization between any of the available calendars and therefore the work was quite unique. The name that was given to the plugin is Excon and we will refer to that from now and on as such.

The methodology chosen for the Excon development was SCRUM. SCRUM is a subset of the so called Agile software development. It is a process framework for agile development and probably the most widely used one. Therefore in the context of SCRUM a set of practices were applied by the development team. The most prominent one is the work division into a number of Sprints. Each Sprint usually has two weeks period cycle but in the Excon case the Sprint was set to one week. A sprint starts with the first day devoted to sprint planning as well as what happened during the last sprint, in case there was one. For the role of coordinator, the team assigned one member with the role of SCRUM member in order to facilitate the whole work flow. During this very important sprint planning meeting the team made a best effort to identify and break down the Stories into clear defined engineering tasks. Main focus was that each member would have a clear understanding of what the goal of the current sprint was and the steps needed in order to reach that. Moreover, to better cope with the changes and increase the quality of the deliverables (mostly for next sprints).

Each day during a sprint started with a Daily SCRUM meeting at 8:15 o' clock strictly. The purpose of this SCRUM meeting was for each team member to describe for the rest team what he/she has done during the last day, various problems that might dealt with and possible need to address/resolve in order to be both be personally better in the future and reporting for the rest of the group so that all together solve it and increase the quality of the work. During that meeting, pair programming pairs created. A rotation after second day was decided among the team so that each pair would have two whole days to work together and deliver its task. Its pair would choose an engineering task from the board and start working with it. During the pair changes, a pair with an unfinished task would explain to the next pair that would take over, about the progress and the difficulties which they dealt with.

The end of each sprint included a Demonstration of the current progress and a retrospective meeting of the team that took somewhere between 2-4 hours. During the retrospective the team reflected to the progress done so far and to what extend the main as well as the sub goals were accomplished. Each member had the chance to express his/her opinion about what went good and what went bad during the last sprint. In this way the team tried to keep the commonly believed good tactics/strategies and eliminate what it was considered inefficient and bad for the progress of the project. This self-improvement method turned out to be very effective and got appreciated by everyone.

3. Methodology

The methodology followed is based on the Self-Governance Developer Framework [2]. In this chapter and more specifically in the first section, the research methodology which was followed by the current project is described, while in the next section (2.2) the evaluation criteria are listed and described.

3.1. Research methodology steps

Conducting the SGD Framework suggested the following research steps

1. Study the SGD Framework
2. Conduct a preliminary feasibility analysis of the project
3. Literature survey
4. Compile weekly results(4 Sprints, compile results on weekly basis)
5. Compile the final project

6. Analyze the project work results and level of completeness

The study of SGD framework summarizes the effort and time spend in order to get an understanding of what it actually means and what the benefits are expected to be. SGD consists of the activities that are conducted as a generalist and the activities that are conducted as a developer.

In the next step, a preliminary feasibility analysis performed in order to create a concrete common understanding among the group of what need to be done, what are the difficulties, the risks, how to deal with these risks, project division in feasible subtasks as well as the tools required in order reach each sub goal at a given planned time frame.

Third step includes a literature survey for mainly identifying the tools and the method previously discussed in order to start dynamically the project and enter into the implementation phase.

At the end of each sprint, the current results were put together so that a complete at that phase product was finished.

The fifth step is the final compile of the whole project, validated through continuous testing and bug fixing. Finally in the final step, the project work analyzed in terms of results in respect to initial goal settings.

Below the figure 1 shows the research method steps.

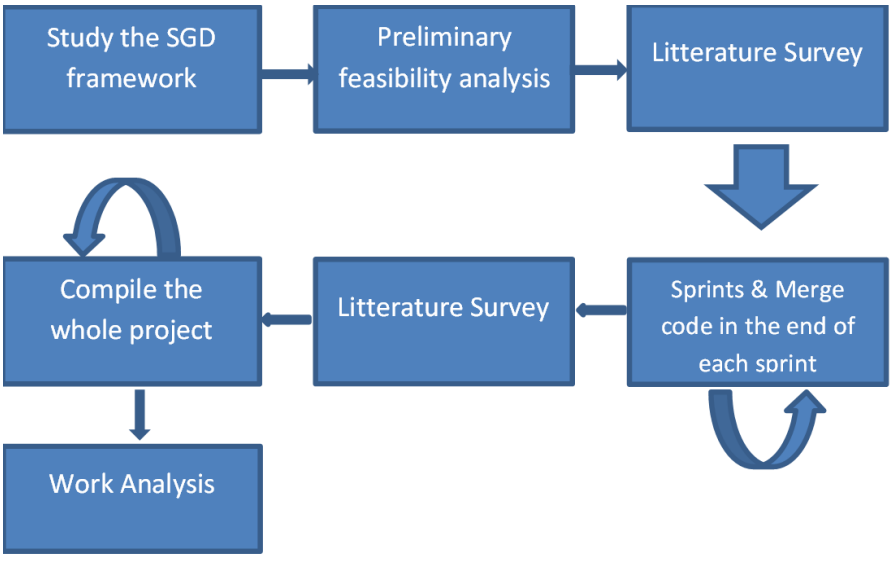

Figure 1: Working process

3.2. Evaluation criteria

In this section the evaluation criteria are to be listed and motivated:

- *Sum of effort spent on each individual activity within the whole project:* A list of developer activities and a list of non-developer activities were present in a form of excel file in which each member reported the time spent during a working day.

- *Distribution of the effort across activity types on an iteration basis*: Each member put a an amount of effort (in hours) during each sprint. Having to put effort in many different activities gives a better level of awareness of what the progress, the stages and the phases the project is going through.

- *Differences of effort spent on various activities in various iterations*: Every iteration resulted in different amount of effort/time

- *Comparison of distribution of effort spent on Non-Developer versus SGD Developer activities*: Quantifying in terms of hours the effort spent from each member on non-developer and developer activities, the team could evaluate the efficiency in respect to the results compared to the expected results, set from the sprint planning/goal. This is possible by identifying a better way to re distribute the time available onto specific tasks more efficient.

4. Results

In the current section the raw data of the individual work are being presented. The total amount of minutes spent in the various parts both as a Skilled Generalist non developer role and as in the role of a (pair) developer are presented in the following tables, table 1 and table 2.

ACTIVITIES THAT I CONDUCT IN THE ROLE OF A SKILLED GENERALIST	
Managing Requirements of a Skilled Generalist Role	TOTAL
REQ 1: Identify requirements	150
REQ 2: Analyze requirements	210
REQ 3: Change requirements	60
REQ 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)	0
REQ 3.2: Make change(s) (time it takes to make changes everywhere in the system)	0
REQ 4: Plan requirements	50
REQ 4.1: Estimate effort	60
REQ 4.2: Prioritize requirements	55
REQ 4.3: Other planning activities related to requirements (specify)	0
REQ 5: Other, specify	990
	1575
Design of a Skilled Generalist Role	
DES 1: Design system or system component (high-level design)'	105
REQ 2: Analyze requirements	80
DES 3: Change design	90
DES 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)	45
DES 3.2: Make change(s) (time it takes to make changes everywhere in the system)	0
DES 4: Other, specify	0
	320
Testing (NON-DEVELOPER Level Testing, acceptance, system, integration) of a Skilled Generalist Role	
TEST 1: Define tests	95
TEST 2: Analyze tests	135
TEST 3: Change tests	0
TEST 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)	0
TEST 3.2: Make change(s) to tests (time it takes to make changes everywhere in the system)	0
TEST 4: Other, specify	0
	230
Project management of a Skilled Generalist Role	
PROJ-MAN 1: Plan project/part of project (iteration)	90
PROJ-MAN 2: Analyze project/ project plan /part of project (iteration)	0
PROJ-MAN 3: Change project plan /part of project plan (iteration)	0
PROJ-MAN 4: Evaluate your project work	130
PROJ-MAN 5: Manage risks	0
PROJ-MAN 5.1: Identify risks	0
PROJ-MAN 5.2: Analyze risks	0
PROJ-MAN 5.3: Manage risks	0
PROJ-MAN 6: Customer-related activities, specify	0
ROJ-MAN 7: Other, specify	0
	220

Table 1: Activities that I conducted in the role of a skilled generalist

SGD ACTIVITIES CONDUCTed IN THE ROLE OF A (Pair)/DEVELOPER	
Preliminary Activities	
PR- 1: Review and agree on the overall or part of the project plan.	175
PR-2: Revise and ensure that the technology to be used is tested and understood.	360
PR-3: Revise and understand any appropriate internal (organizational) and external standard(s).	0
PR-4: Learn/relearn the organizational implementation and unit (developer) testing way of working.	0
PR-5: Review and revise your personal implementation and unit (developer) testing way of working.	0
PR-6: Other, specify	0
	535
Planning Activities of a Developer Role	
PL-1: Review the requirement(s) for the unit(s) to be developed.	10
PL-2: Prepare (make) and/or review the design specification(s) for the unit(s) to be developed.	0
PL-3: Resolve unclear questions and uncertainties.	50
PL-4: Determine and document your implementation and unit (developer) testing goals.	40
PL-5: Determine your implementation and unit (developer) testing strategy.	20
PL-6: Determine appropriate implementation and testing practices.	0
PL-7: Identify standards to be used for meeting your goals.	60
PL-8: Set your own personal deadlines to be met during your implementation and unit (developer) testing work.	0
PL-9: Estimate effort and resources required for carrying out your work.	45
PL-10: Schedule your work.	0
PL-11: Review your implementation and unit (developer) testing plan to ensure that it is realistic and achievable.	0
PL-12: Identify risks related to your plan.	0
PL-13: Plan for managing any identified risks.	0
PL-14: Other, specify	240
	465
Preparatory Activities of a Developer Role	
P-1: Prepare(make) and/or review your low-level design(s) of the code to be written or changed.	30
P-2: Prepare (make) an impact analysis of your low-level design(s).	60
P-3: Determine the types of unit (developer) test cases and their order.	0
P-4: Create and/or revise your unit (developer) test case base.	0
P-5: Revise the existing unit (developer) regression test base, if relevant.	0
P-6: Create or modify stubs and drivers, if required.	0
P-7: Prepare your unit (developer) testing environment and check whether it is appropriate for you work.	0
P-8: Other, specify	0
	90
Coding Activities of a Developer Role	
C-1: Write/rewrite your code.	3610
C-2: Compile/ recompile your code as required.	455
C-3: Make notes on your compilation errors, if necessary.	0
C-4: Make notes on your defects	0
C-5: Other, specify	0
	4065
Unit Testing Activities of a Developer Role	
T-1: Check whether the unit (developer) test case base meets the given requirements and design.	150
T-2: Check whether the unit (developer) regression test base meets the given requirements and design.	0
T-3: Remedy requirements problems in your unit (developer) regression and/or test cases base, if any.	0
T-4: Perform dynamic testing by executing code.	590
T-5: Perform static (human) testing by reviewing your code.	395
T-6: Record/write down test results.	90
T-7: Other, specify	0
	1225
Evaluative Activities of a Developer Role	
E-1: Analyze your unit (developer) testing results.	45
E-2: Depending on the unit (developer) testing results, determine your next step(s).	15
E-3: Other, specify	0
	60
Debugging Activities of a Developer Role	
D-1: Identify the source of (an) error(s).	550
D-2: Determine solution(s) for eliminating the sources of error(s).	465
D-3: Other, specify	0
	1015
Self-Assessment Activities (Document aside your self-assessment results)	
A-1: Assess your own development work.	0
A-2: Identify causes of your mistakes.	35
A-3: Identify improvement areas in your own way of working.	55
A-4: Other, specify	0
	90
Delivery of a Developer Role	
S-2: Deliver your code.	150
S-3: Other, specify	0
	150

Table 2: Activities that I conducted in the role of a (Pair)/Developer

4.1. Individual Activity within the whole project

In the table one the total sum of the minutes of various activities spent is 2345 minutes of individual work as a non-developer. The most time in this part was spent in the daily SCRUM meetings as well as the middle small meeting that the team decided to have occasionally, but rather often.

For the design and the testing (non-developer) as a skilled generalist role the total amount of time spent was 320 and 230 minutes respectively.

In table 2 the total amount of time spent working as developer either alone or as a pair in various activities is 7545 min. The time spent for coding takes the majority of the time, namely 4065 minutes. Unit testing or testing part of the code tool 1225 min and the debugging phases 1015 min. Moreover the preliminary activities together with the planning activities took in together 1000 min (almost 50% each). The rest of the time which account for a few hours in total, was spent in various activities such as preparatory, self-assessment and delivery of the code.

4.2. Distribution of the effort across Activity types on an Iteration basis

The whole project was divided into four weekly sprints. Hence in this section a distribution of the time spent in some key parts of the project on an iteration basis (during the four sprints) in the various activities are presented. As expected the amount of time spent during each sprint was almost the same as one sprint was one week of 8 hours work per day.

Having a closer look to some of the activities it can be observed that there are some interesting distributions and trends. Figure 2, Figure 3 and Figure 4 are referring to the time spent through the four sprints as a developer.

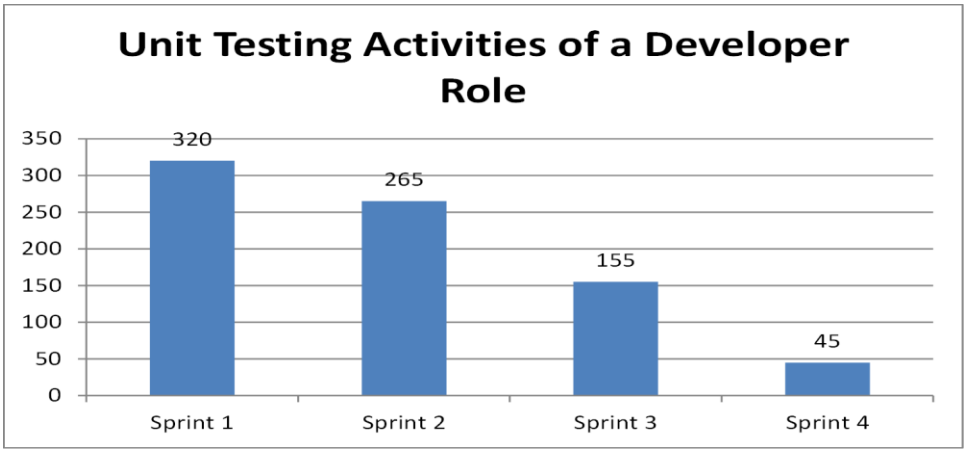

Figure 2: Time spent on testing activities

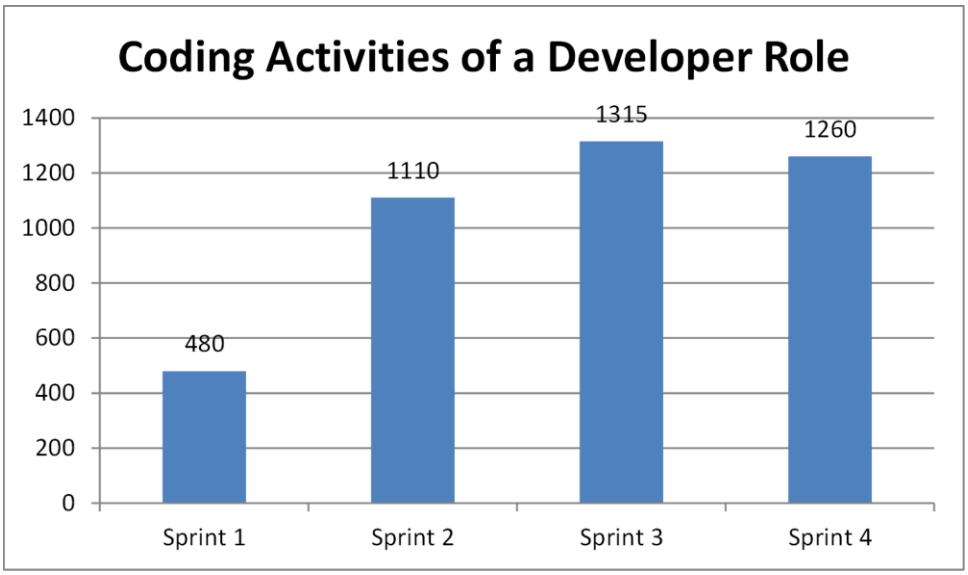

Figure 3: Work effort for coding through the four iterations

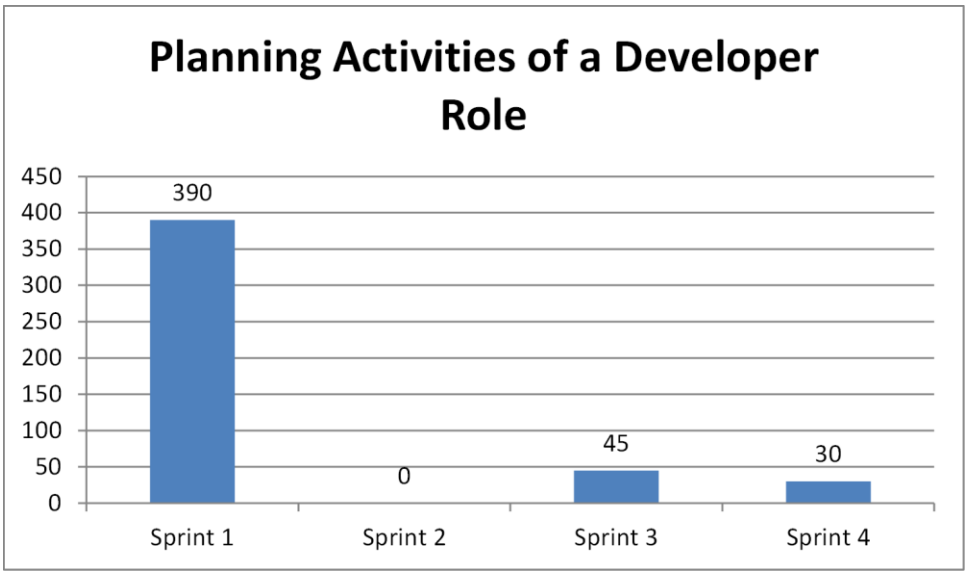

Figure 4: Time spent on planning activities during the four iterations

A more specific result regarding the developer role activities is about how the time was spent during the four sprints for performing dynamic testing by executing code. Figure 5 below show the distribution:

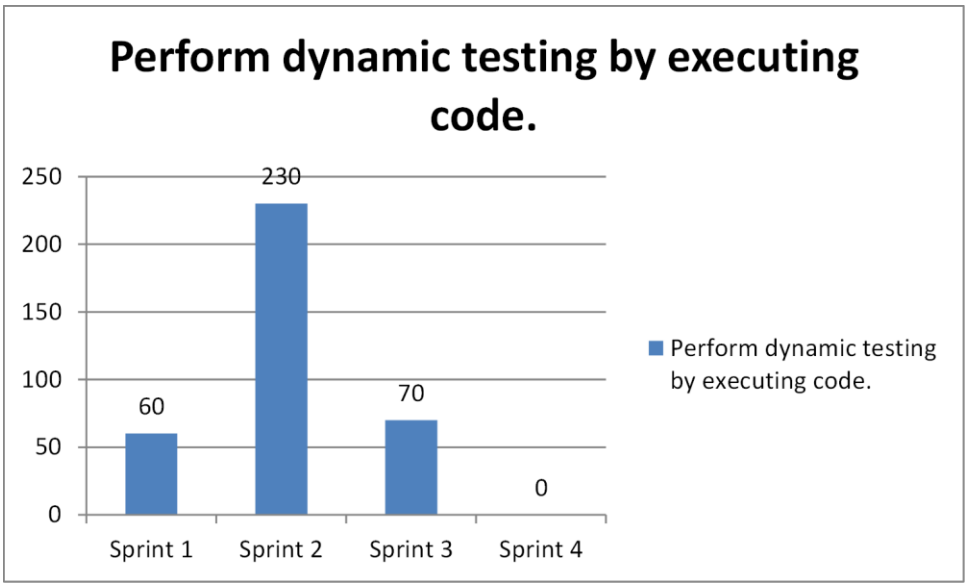

Figure 5: Perform dynamic testing by executing code time distribution over the four iterations

Further down in figures 6, 7 some interesting results in the non-developer role are demonstrated.

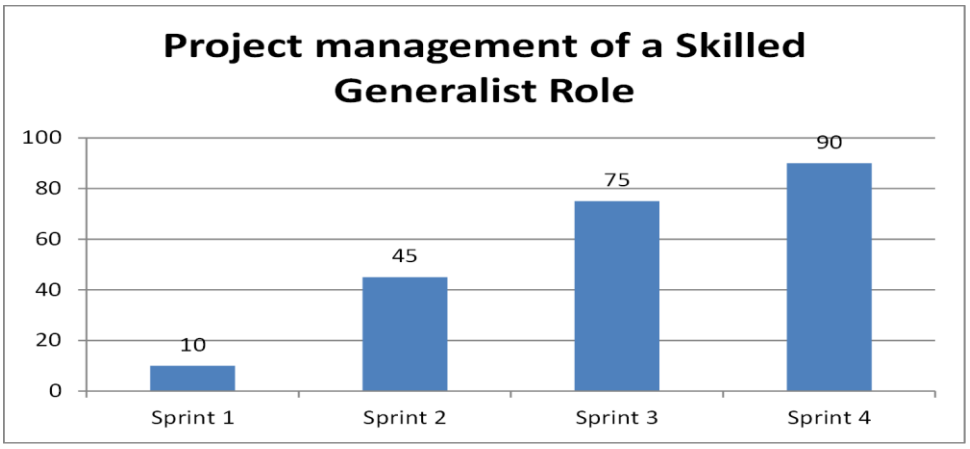

Figure 6: Time spent in Non-developer role for project management

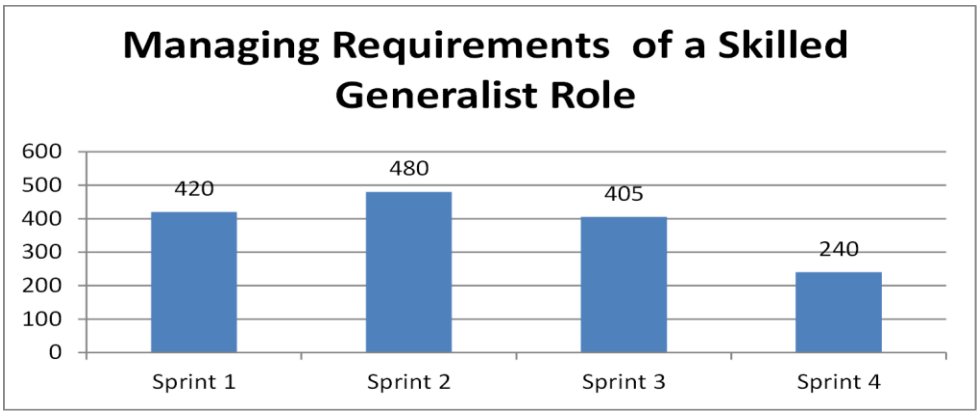

Figure 7: Time spent in Non-developer role for Managing requirements

4.3. Differences of effort spent on various Activities in various Iterations

It is observed that there is an obvious distribution of approximately 75 % for work put on developer activities in four different iterations. Time spent for coding during the first sprint is considerably lower compared to the next three. Time spent for testing gradually lowered from spring to spring.

Moreover, a best effort for planning activities during the first sprint is also apparent (see Figure 5) as the time spent in spring 1 is almost 85% of the total time spent for that activity throughout the whole project.

4.4. Comparison of distribution of effort spent on a Non-Developer VS SGD developer Activities

The work effort I conducted in activities in the role of a skilled generalist and as a pair or alone developer for the four iterations is presented in the figure below:

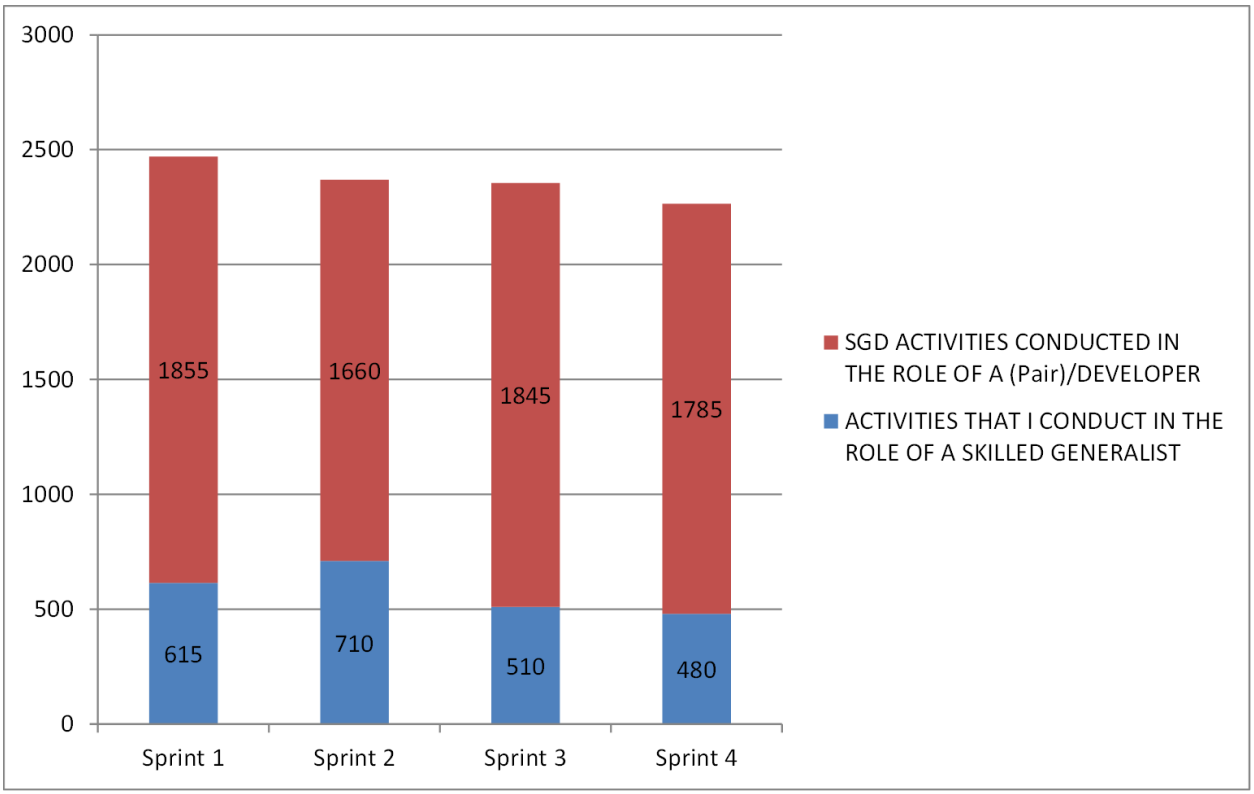

Figure 8: SGD Developer VS Non-Developer work effort distribution over the four iterations

As it is observed, the amount of time spent for coding activities during the four iterations is approximately 75% of the total time. The rest was devoted to non-developer activities

5. Result analysis

5.1. Analysis

In total, as mentioned previously the total amount of time spent on SGD activities took about the three quarters of the whole project. In particular pure coding, testing code as well as debugging took most of the time from the SGD activities. This is quite normal and expected for two reasons. The first one because the project itself is a software development project and the second one is due to the fact that the resources about how to create the plug in were limited. This lead to continuous experimentation and a lot of time put in trying and error in coding. It is also shown that the time spent for SGD activities were almost evenly distributed throughout the four sprints. However, looking a bit closer in figure 3, it can be observed that during the first sprint the time spent on pure coding is significantly less compared to the next three sprints. The reason for that is the fact that during the first sprint a lot of time was spent for preparation, literature survey and planning on how we were going to proceed in terms of structure of the project as well as starting with some familiarization tutorials.

On figure 4 it is shown that an initial big effort to have as good preparation as possible for the developer activities is made. A good preparation during the first sprint was quite important. The whole team was unexperienced in developing software, working in terms of SCRUM and much other parameters were unknown. Thus the first sprint was at the same time a pilot sprint where each member would find him/herself in the project, and a preparation of the next three sprints.

Moreover, an important part of developing, that of testing, seems to not follow an ordered pattern (figure 5). During the first sprint the actual testing started only after the three first days as before there was nothing to test. During that time and during the second sprint, the time spent on testing and executing code was increased. However during the third and fourth sprint it did dropped significantly. One explanation for this can be the fact that at some points some changes in the project structure needed to be made. This did not allow for much testing new or existing code that already worked. Moreover, the testing part somehow came second to priority when the group had to change the database that it was chosen up to that time.

In non-developer activities it is observed that the amount of time spent on managing requirements of a skilled generalist sum up to the 65% of the total amount of time spent only on the non-developer activities. This is also logical as much time of this category of activities was spent on SCRUM meetings (45% of total non-developer activities), sprint planning at the beginning of every sprint as well as the retrospective and demonstration day at the end of each sprint.

On figure 6 it is shown how the amount of time spent for the project management gradually increased from sprint one throughout the finish of the project. This is also logical as in the beginning there is not a product realization yet, but through time the need to manage the current progress and decide how to proceed becomes apparent. While in the end of the sprint 3 the project should have been somewhere close to the end, the management time spent could be expected lowered. However in this case there was a big matter on how to proceed in terms of which database it would be used. The database that the team used from the beginning was MySQL while the database that the company was interested in using was

another one. The question was whether to migrate everything to the new database or to optimize the current one.

6. Conclusions

The project from the beginning was challenging due to limited prior knowledge and resources available both on synchronization of calendars as well as creating plugins for the Confluence platform. Lack of prior experience on working on SCRUM based terms was also something that delayed us in the beginning as it was considered as an experimentation period nonetheless. Testing is something that could have been executed better. Despite the fact the team was aware from the beginning about creating test matrices and methodologies, which it did, as the time passed by, the focus was directed mostly in making things done. This is not the ideal procedure and may cause frustration and loose even more valuable time instead of saving. The reason is that a good testing prevents from continuing to a later stage of the development process with code that is not thoroughly tested under any circumstance. Future bugs that are not yet apparent may and could be avoided by testing code more often. Thus, during the third and fourth sprint the testing was not optimal and this is something that could be done better in future projects. Therefore, a better distribution of the testing responsibilities should be taken into account in future similar projects.

Despite the fact that I did not have a prior experience in working with IT project in an iterative method like SCRUM, the advantages are directly understood and appreciated. I consider that, given the lack of prior experience in where to put the necessary effort in terms of working hours, the whole execution was sucessful based on what the result was in respect to the effort distribution.

7. Future work and improvements

Some improvements closely related to the project, could possibly be the tighter collaboration with the company that has requested the creation of a solution. In this case, while there was always the feeling and the guarantees that help from Betsson was always available, there were only a few moments that the development team actually asked for it. A valid motivation for doing so could be, avoiding putting effort on things that experienced SRCUM teams do not do or more likely distribute the total amount of effort in different activities of what I actually did. For example, the problem that could have been solved earlier in such case, would be the database that was chosen and turned out not to be the one that Betsson was using. This would have probably saved us from the time spent during the last sprint for migrating the project from mySQL to Microsoft SQL. In such case much of the time spent during the fourth sprint on debugging, migration of the database and management of the project could have easily been used for further testing and creating a more complete test environment in order to test the final product in real working environment.

In terms of working agile in the frame of SCRUM, improvements are obviously going to come with continuously practicing and being member of such teams. Given the obvious advantages of this iterative method, there is also a feeling that with experience, wiser and more to the point decisions are going to be made by continuously developing as member of SCRUM teams.

References

[1] https://www.scrumalliance.org/why-scrum

[2] Mira Kajko-Matsson, Gudrun Jeppesen, *Self-Governance Developer Framework*

APPENDIX

TOTAL TIME SPENT ON ACTIVITIES PER ITERATION

ACTIVITIES THAT I CONDUCT IN THE ROLE OF A SKILLED GENERALIST				
Managing Requirements of a Skilled Generalist Role	**Sprint 1**	**Sprint 2**	**Sprint 3**	**Sprint 4**
REQ 1: Identify requirements	60	60	30	0
REQ 2: Analyze requirements	60	120	30	0
REQ 3: Change requirements	0	0	60	0
REQ 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)	0	0	0	0
REQ 3.2: Make change(s) (time it takes to make changes everywhere in the system)	0	0	0	0
REQ 4: Plan requirements	0	20	30	0
REQ 4.1: Estimate effort	30	30	0	0
REQ 4.2: Prioritize requirements	30	10	15	0
REQ 4.3: Other planning activities related to requirements (specify)	0	0	0	0
REQ 5: Other, specify	240	240	240	240
	420	**480**	**405**	**240**
Design of a Skilled Generalist Role				
DES 1: Design system or system component (high-level design)'	0	105	0	0
REQ 2: Analyze requirements	0	20	0	60
DES 3: Change design	60	30	0	0
DES 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)	15	30	0	0
DES 3.2: Make change(s) (time it takes to make changes everywhere in the system)	0	0	0	0
DES 4: Other, specify	0	0	0	0
	75	**185**	**0**	**60**
Testing (NON-DEVELOPER Level Testing, acceptance, system, integration) of a Skilled Generalist Role				
TEST 1: Define tests	50	0	0	45
TEST 2: Analyze tests	60	0	30	45
TEST 3: Change tests	0	0	0	0
TEST 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)	0	0	0	0
TEST 3.2: Make change(s) to tests (time it takes to make changes everywhere in the system)	0	0	0	0
TEST 4: Other, specify	0	0	0	0
	110	**0**	**30**	**90**
Project management of a Skilled Generalist Role				
PROJ-MAN 1: Plan project/part of project (iteration)	0	15	15	60
PROJ-MAN 2: Analyze project/ project plan /part of project (iteration)	0	0	0	0
PROJ-MAN 3: Change project plan /part of project plan (iteration)	0	0	0	0
PROJ-MAN 4: Evaluate your project work	10	30	60	30
PROJ-MAN 5: Manage risks	0	0	0	0
PROJ-MAN 5.1: Identify risks	0	0	0	0
PROJ-MAN 5.2: Analyze risks	0	0	0	0
PROJ-MAN 5.3: Manage risks	0	0	0	0
PROJ-MAN 6: Customer-related activities, specify	0	0	0	0
ROJ-MAN 7: Other, specify	0	0	0	0
	10	**45**	**75**	**90**
	615	**710**	**510**	**480**

SGD ACTIVITIES CONDUCTED IN THE ROLE OF A (Pair)/DEVELOPER				
Preliminary Activities				
PR- 1: Review and agree on the overall or part of the project plan.	100	30	15	30
PR-2: Revise and ensure that the technology to be used is tested and understood.	215	105	40	0
PR-3: Revise and understand any appropriate internal (organizational) and external standard(s).	0	0	0	0
PR-4: Learn/relearn the organizational implementation and unit (developer) testing way of working.	0	0	0	0
PR-5: Review and revise your personal implementation and unit (developer) testing way of working.	0	0	0	0
PR-6: Other, specify	0	0	0	0
	315	**135**	**55**	**30**
Planning Activities of a Developer Role				
PL-1: Review the requirement(s) for the unit(s) to be developed.	10	0	0	0
PL-2: Prepare (make) and/or review the design specification(s) for the unit(s) to be developed.	0	0	0	0
PL-3: Resolve unclear questions and uncertainties.	20	0	30	0
PL-4: Determine and document your implementation and unit (developer) testing goals.	40	0	0	0
PL-5: Determine your implementation and unit (developer) testing strategy.	20	0	0	0
PL-6: Determine appropriate implementation and testing practices.	0	0	0	0
PL-7: Identify standards to be used for meeting your goals.	60	0	0	0
PL-8: Set your own personal deadlines to be met during your implementation and unit (developer) testing work.	0	0	0	0
PL-9: Estimate effort and resources required for carrying out your work.	0	0	15	30
PL-10: Schedule your work.	0	0	0	0
PL-11: Review your implementation and unit (developer) testing plan to ensure that it is realistic and achievable.	0	0	0	0
PL-12: Identify risks related to your plan.	0	0	0	0
PL-13: Plan for managing any identified risks.	0	0	0	0
PL-14: Other, specify	240	0	0	0
	390	**0**	**45**	**30**
Preparatory Activities of a Developer Role				
P-1: Prepare(make) and/or review your low-level design(s) of the code to be written or changed.	0	30	0	0
P-2: Prepare (make) an impact analysis of your low-level design(s).	60	0	0	0
P-3: Determine the types of unit (developer) test cases and their order.	0	0	0	0
P-4: Create and/or revise your unit (developer) test case base.	0	0	0	0
P-5: Revise the existing unit (developer) regression test base, if relevant.	0	0	0	0
P-6: Create or modify stubs and drivers, if required.	0	0	0	0
P-7: Prepare your unit (developer) testing environment and check whether it is appropriate for you work.	0	0	0	0
P-8: Other, specify	0	0	0	0
	60	**30**	**0**	**0**
Coding Activities of a Developer Role				
C-1: Write/rewrite your code.	420	1090	1220	980
C-2: Compile/ recompile your code as required.	60	20	95	280
C-3: Make notes on your compilation errors, if necessary.	0	0	0	0
C-4: Make notes on your defects	0	0	0	0
C-5: Other, specify	0	0	0	0
	480	**1110**	**1315**	**1260**
Unit Testing Activities of a Developer Role				
T-1: Check whether the unit (developer) test case base meets the given requirements and design.	150	0	0	0
T-2: Check whether the unit (developer) regression test base meets the given requirements and design.	0	0	0	0
T-3: Remedy requirements problems in your unit (developer) regression and/or test cases base, if any.	0	0	0	0
T-4: Perform dynamic testing by executing code.	60	230	70	0
T-5: Perform static (human) testing by reviewing your code.	20	35	85	45
T-6: Record/write down test results.	90	0	0	0
T-7: Other, specify	0	0	0	0
	320	**265**	**155**	**45**
Evaluative Activities of a Developer Role				
E-1: Analyze your unit (developer) testing results.	0	0	45	0
E-2: Depending on the unit (developer) testing results, determine your next step(s).	0	0	15	0
E-3: Other, specify	0	0	0	0
	0	0	60	0
Debugging Activities of a Developer Role				
D-1: Identify the source of (an) error(s).	140	45	125	240
D-2: Determine solution(s) for eliminating the sources of error(s).	90	75	120	180
D-3: Other, specify	0	0	0	0
	230	**120**	**245**	**420**
Self-Assessment Activities (Document aside your self-assessment results)				
A-1: Assess your own development work.	0	0	0	0
A-2: Identify causes of your mistakes.	20	0	15	0
A-3: Identify improvement areas in your own way of working.	40	0	15	0
A-4: Other, specify	0	0	0	0
	60	**0**	**30**	**0**
Delivery of a Developer Role				
S-2: Deliver your code.	0	30	40	80
S-3: Other, specify	0	0	0	0
	2470	**2400**	**2455**	**2345**

SPRINT 1

ACTIVITIES THAT I CONDUCT IN THE ROLE OF A SKILLED GENERALIST							
Managing Requirements of a Skilled Generalist Role							
REQ 1: Identify requirements				60			**60**
REQ 2: Analyze requirements				60			**60**
REQ 3: Change requirements							**0**
REQ 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)							**0**
REQ 3.2: Make change(s) (time it takes to make changes everywhere in the system)							**0**
REQ 4: Plan requirements							**0**
REQ 4.1: Estimate effort				30			**30**
REQ 4.2: Prioritize requirements				30			**30**
REQ 4.3: Other planning activities related to requirements (specify)							**0**
REQ 5: Other, specify (SCRUM Meetings)	60	60			60	60	**240**
Design of a Skilled Generalist Role							
DES 1: Design system or system component (high-level design)'							**0**
REQ 2: Analyze requirements							**0**
DES 3: Change design						60	**60**
DES 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)						15	**15**
DES 3.2: Make change(s) (time it takes to make changes everywhere in the system)							**0**
DES 4: Other, specify							**0**
Testing (NON-DEVELOPER Level Testing, acceptance, system, integration) of a Skilled Generalist Role							
TEST 1: Define tests		50					**50**
TEST 2: Analyze tests		60					**60**
TEST 3: Change tests							**0**
TEST 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)							**0**
TEST 3.2: Make change(s) to tests (time it takes to make changes everywhere in the system)							**0**
TEST 4: Other, specify							**0**
Project management of a Skilled Generalist Role							
PROJ-MAN 1: Plan project/part of project (iteration)							**0**
PROJ-MAN 2: Analyze project/ project plan /part of project (iteration)							**0**
PROJ-MAN 3: Change project plan /part of project plan (iteration)							**0**
PROJ-MAN 4: Evaluate your project work	10						**10**
PROJ-MAN 5: Manage risks							**0**
PROJ-MAN 5.1: Identify risks							**0**
PROJ-MAN 5.2: Analyze risks							**0**
PROJ-MAN 5.3: Manage risks							**0**
PROJ-MAN 6: Customer-related activities, specify							**0**
ROJ-MAN 7: Other, specify							**0**
							615

SPRINT 2

ACTIVITIES THAT I CONDUCT IN THE ROLE OF A SKILLED GENERALIST											
Managing Requirements of a Skilled Generalist Role				Sprint Planning							
REQ 1: Identify requirements				60							**60**
REQ 2: Analyze requirements				120							**120**
REQ 3: Change requirements											**0**
REQ 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)											**0**
REQ 3.2: Make change(s) (time it takes to make changes everywhere in the system)											**0**
REQ 4: Plan requirements				20							**20**
REQ 4.1: Estimate effort				30							**30**
REQ 4.2: Prioritize requirements				10							**10**
REQ 4.3: Other planning activities related to requirements (specify)											**0**
REQ 5: Other, specify	60		60			60		60			**240**
											0
Design of a Skilled Generalist Role											**0**
DES 1: Design system or system component (high-level design)'			60					45			**105**
REQ 2: Analyze requirements				20							**20**
DES 3: Change design						30					**30**
DES 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)			30								**30**
DES 3.2: Make change(s) (time it takes to make changes everywhere in the system)											**0**
DES 4: Other, specify											**0**
											0
Testing (NON-DEVELOPER Level Testing, acceptance, system, integration) of a Skilled Generalist Role											**0**
TEST 1: Define tests											**0**
TEST 2: Analyze tests											**0**
TEST 3: Change tests											**0**
TEST 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)											**0**
TEST 3.2: Make change(s) to tests (time it takes to make changes everywhere in the system)											**0**
TEST 4: Other, specify											**0**
											0
Project management of a Skilled Generalist Role											**0**
PROJ-MAN 1: Plan project/part of project (iteration)								15			**15**
PROJ-MAN 2: Analyze project/ project plan /part of project (iteration)											**0**
PROJ-MAN 3: Change project plan /part of project plan (iteration)											**0**
PROJ-MAN 4: Evaluate your project work	20			10							**30**
PROJ-MAN 5: Manage risks											**0**
PROJ-MAN 5.1: Identify risks											**0**
PROJ-MAN 5.2: Analyze risks											**0**
PROJ-MAN 5.3: Manage risks											**0**
PROJ-MAN 6: Customer-related activities, specify											**0**
ROJ-MAN 7: Other, specify											**0**

SGD ACTIVITIES CONDUCTed IN THE ROLE OF A (Pair)/DEVELOPER										
Preliminary Activities										
PR- 1: Review and agree on the overall or part of the project plan.					30					**30**
PR-2: Revise and ensure that the technology to be used is tested and understood.								105		**105**
PR-3: Revise and understand any appropriate internal (organizational) and external standard(s).										**0**
PR-4: Learn/relearn the organizational implementation and unit (developer) testing way of working.										**0**
PR-5: Review and revise your personal implementation and unit (developer) testing way of working.										**0**
PR-6: Other, specify										**0**
										0
Planning Activities of a Developer Role										**0**
PL-1: Review the requirement(s) for the unit(s) to be developed.										**0**
PL-2: Prepare (make) and/or review the design specification(s) for the unit(s) to be developed.										**0**
PL-3: Resolve unclear questions and uncertainties.										**0**
PL-4: Determine and document your implementation and unit (developer) testing goals.										**0**
PL-5: Determine your implementation and unit (developer) testing strategy.										**0**
PL-6: Determine appropriate implementation and testing practices.										**0**
PL-7: Identify standards to be used for meeting your goals.										**0**
PL-8: Set your own personal deadlines to be met during your implementation and unit (developer) testing work.										**0**
PL-9: Estimate effort and resources required for carrying out your work.										**0**
PL-10: Schedule your work.										**0**
PL-11: Review your implementation and unit (developer) testing plan to ensure that it is realistic and achievable.										**0**
PL-12: Identify risks related to your plan.										**0**
PL-13: Plan for managing any identified risks.										**0**
PL-14: Other, specify										**0**
										0
Preparatory Activities of a Developer Role										**0**
P-1: Prepare(make) and/or review your low-level design(s) of the code to be written or changed.								30		**30**
P-2: Prepare (make) an impact analysis of your low-level design(s).										**0**
P-3: Determine the types of unit (developer) test cases and their order.										**0**
P-4: Create and/or revise your unit (developer) test case base.										**0**
P-5: Revise the existing unit (developer) regression test base, if relevant.										**0**
P-6: Create or modify stubs and drivers, if required.										**0**
P-7: Prepare your unit (developer) testing environment and check whether it is appropriate for you work.										**0**
P-8: Other, specify										**0**
										0
Coding Activities of a Developer Role										**0**
C-1: Write/rewrite your code.	400		300			240		150		**1090**
C-2: Compile/ recompile your code as required.						20				**20**
C-3: Make notes on your compilation errors, if necessary.										**0**
C-4: Make notes on your defects										**0**
C-5: Other, specify										**0**
										0
Unit Testing Activities of a Developer Role										**0**
T-1: Check whether the unit (developer) test case base meets the given requirements and design.										**0**
T-2: Check whether the unit (developer) regression test base meets the given requirements and design.										**0**
T-3: Remedy requirements problems in your unit (developer) regression and/or test cases base, if any.										**0**
T-4: Perform dynamic testing by executing code.					180	25		15		**220**
T-5: Perform static (human) testing by reviewing your code.						20		15		**35**
T-6: Record/write down test results.										**0**
T-7: Other, specify										**0**
										0
Evaluative Activities of a Developer Role										**0**
E-1: Analyze your unit (developer) testing results.										**0**
E-2: Depending on the unit (developer) testing results, determine your next step(s).										**0**
E-3: Other, specify										**0**
										0
Debugging Activities of a Developer Role										**0**
D-1: Identify the source of (an) error(s).						30		15		**45**
D-2: Determine solution(s) for eliminating the sources of error(s).						60		15		**75**
D-3: Other, specify										**0**
										0
Self-Assessment Activities (Document aside your self-assessment results)										**0**
A-1: Assess your own development work.										**0**
A-2: Identify causes of your mistakes.										**0**
A-3: Identify improvement areas in your own way of working.										**0**
A-4: Other, specify										**0**
										0
Delivery of a Developer Role										**0**
S-2: Deliver your code.								30		**30**
S-3: Other, specify										**0**
									TOTAL	**2390**

SPRINT 3

ACTIVITIES THAT I CONDUCT IN THE ROLE OF A SKILLED GENERALIST							
Managing Requirements of a Skilled Generalist Role			Retrospective/Demop				
REQ 1: Identify requirements				30			**30**
REQ 2: Analyze requirements				30			**30**
REQ 3: Change requirements			60				**60**
REQ 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)							**0**
REQ 3.2: Make change(s) (time it takes to make changes everywhere in the system)							**0**
REQ 4: Plan requirements			30				**30**
REQ 4.1: Estimate effort							**0**
REQ 4.2: Prioritize requirements				15			**15**
REQ 4.3: Other planning activities related to requirements (specify)							**0**
REQ 5: Other, specify	60	60		60	60		**240**
							0
Design of a Skilled Generalist Role							**0**
DES 1: Design system or system component (high-level design)'							**0**
REQ 2: Analyze requirements							**0**
DES 3: Change design							**0**
DES 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)							**0**
DES 3.2: Make change(s) (time it takes to make changes everywhere in the system)							**0**
DES 4: Other, specify							**0**
							0
Testing (NON-DEVELOPER Level Testing, acceptance, system, integration) of a Skilled Generalist Role							**0**
TEST 1: Define tests							**0**
TEST 2: Analyze tests	30						**30**
TEST 3: Change tests							**0**
TEST 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)							**0**
TEST 3.2: Make change(s) to tests (time it takes to make changes everywhere in the system)							**0**
TEST 4: Other, specify							**0**
							0
Project management of a Skilled Generalist Role							**0**
PROJ-MAN 1: Plan project/part of project (iteration)				15			**15**
PROJ-MAN 2: Analyze project/ project plan /part of project (iteration)							**0**
PROJ-MAN 3: Change project plan /part of project plan (iteration)							**0**
PROJ-MAN 4: Evaluate your project work			60				**60**
PROJ-MAN 5: Manage risks							**0**
PROJ-MAN 5.1: Identify risks							**0**
PROJ-MAN 5.2: Analyze risks							**0**
PROJ-MAN 5.3: Manage risks							**0**
PROJ-MAN 6: Customer-related activities, specify							**0**
RQJ-MAN 7: Other, specify							**0**

SGD ACTIVITIES CONDUCTed IN THE ROLE OF A (Pair)/DEVELOPER							
Preliminary Activities							
PR- 1: Review and agree on the overall or part of the project plan.			15				**15**
PR-2: Revise and ensure that the technology to be used is tested and understood.			40				**40**
PR-3: Revise and understand any appropriate internal (organizational) and external standard(s).							**0**
PR-4: Learn/relearn the organizational implementation and unit (developer) testing way of working.							**0**
PR-5: Review and revise your personal implementation and unit (developer) testing way of working.							**0**
PR-6: Other, specify							**0**
Planning Activities of a Developer Role							
PL-1: Review the requirement(s) for the unit(s) to be developed.							**0**
PL-2: Prepare (make) and/or review the design specification(s) for the unit(s) to be developed.							**0**
PL-3: Resolve unclear questions and uncertainties.			30				**30**
PL-4: Determine and document your implementation and unit (developer) testing goals.							**0**
PL-5: Determine your implementation and unit (developer) testing strategy.							**0**
PL-6: Determine appropriate implementation and testing practices.							**0**
PL-7: Identify standards to be used for meeting your goals.							**0**
PL-8: Set your own personal deadlines to be met during your implementation and unit (developer) testing work.							**0**
PL-9: Estimate effort and resources required for carrying out your work.					15		**15**
PL-10: Schedule your work.							**0**
PL-11: Review your implementation and unit (developer) testing plan to ensure that it is realistic and achievable.							**0**
PL-12: Identify risks related to your plan.							**0**
PL-13: Plan for managing any identified risks.							**0**
PL-14: Other, specify							**0**
Preparatory Activities of a Developer Role							
P-1: Prepare(make) and/or review your low-level design(s) of the code to be written or changed.							**0**
P-2: Prepare (make) an impact analysis of your low-level design(s).							**0**
P-3: Determine the types of unit (developer) test cases and their order.							**0**
P-4: Create and/or revise your unit (developer) test case base.							**0**
P-5: Revise the existing unit (developer) regression test base, if relevant.							**0**
P-6: Create or modify stubs and drivers, if required.							**0**
P-7: Prepare your unit (developer) testing environment and check whether it is appropriate for you work.							**0**
P-8: Other, specify							**0**
Coding Activities of a Developer Role							
C-1: Write/rewrite your code.	270	330		220	300		**1120**
C-2: Compile/ recompile your code as required.	15		20	60			**95**
C-3: Make notes on your compilation errors, if necessary.							**0**
C-4: Make notes on your defects							**0**
C-5: Other, specify							**0**
Unit Testing Activities of a Developer Role							
T-1: Check whether the unit (developer) test case base meets the given requirements and design.							**0**
T-2: Check whether the unit (developer) regression test base meets the given requirements and design.							**0**
T-3: Remedy requirements problems in your unit (developer) regression and/or test cases base, if any.							**0**
T-4: Perform dynamic testing by executing code.	30			20	20		**70**
T-5: Perform static (human) testing by reviewing your code.	45		20		20		**85**
T-6: Record/write down test results.							**0**
T-7: Other, specify							**0**
Evaluative Activities of a Developer Role							
E-1: Analyze your unit (developer) testing results.			30		15		**45**
E-2: Depending on the unit (developer) testing results, determine your next step(s).					15		**15**
E-3: Other, specify							**0**
							0
Debugging Activities of a Developer Role							**0**
D-1: Identify the source of (an) error(s).		60	30	20	15		**125**
D-2: Determine solution(s) for eliminating the sources of error(s).		60		30	30		**120**
D-3: Other, specify							**0**
Self-Assessment Activities (Document aside your self-assessment results)							
A-1: Assess your own development work.							**0**
A-2: Identify causes of your mistakes. (done as group)			15				**15**
A-3: Identify improvement areas in your own way of working.(done as group)			15				**15**
A-4: Other, specify							**0**
							0
Delivery of a Developer Role							**0**
S-2: Deliver your code.	30				10		**40**
S-3: Other, specify							**0**
						TOTAL	**2355**

SPRINT 4

ACTIVITIES THAT I CONDUCT IN THE ROLE OF A SKILLED GENERALIST						
Managing Requirements of a Skilled Generalist Role						
REQ 1: Identify requirements						**0**
REQ 2: Analyze requirements						**0**
REQ 3: Change requirements						**0**
REQ 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)						**0**
REQ 3.2: Make change(s) (time it takes to make changes everywhere in the system)						**0**
REQ 4: Plan requirements						**0**
REQ 4.1: Estimate effort						**0**
REQ 4.2: Prioritize requirements						**0**
REQ 4.3: Other planning activities related to requirements (specify)						**0**
REQ 5: Other, specify	30	30	60	60	60	**240**
Design of a Skilled Generalist Role						
DES 1: Design system or system component (high-level design)'						**0**
REQ 2: Analyze requirements				60		**60**
DES 3: Change design						**0**
DES 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)						**0**
DES 3.2: Make change(s) (time it takes to make changes everywhere in the system)						**0**
DES 4: Other, specify						**0**
Testing (NON-DEVELOPER Level Testing, acceptance, system, integration) of a Skilled Generalist Role						
TEST 1: Define tests		15	30			**45**
TEST 2: Analyze tests		15	30			**45**
TEST 3: Change tests						**0**
TEST 3.1: Identify impact of change (time it takes to identify changes everywhere in the system)						**0**
TEST 3.2: Make change(s) to tests (time it takes to make changes everywhere in the system)						**0**
TEST 4: Other, specify						**0**
Project management of a Skilled Generalist Role						
PROJ-MAN 1: Plan project/part of project (iteration)				60		**60**
PROJ-MAN 2: Analyze project/ project plan /part of project (iteration)						**0**
PROJ-MAN 3: Change project plan /part of project plan (iteration)						**0**
PROJ-MAN 4: Evaluate your project work			30			**30**
PROJ-MAN 5: Manage risks						**0**
PROJ-MAN 5.1: Identify risks						**0**
PROJ-MAN 5.2: Analyze risks						**0**
PROJ-MAN 5.3: Manage risks						**0**
PROJ-MAN 6: Customer-related activities, specify						**0**
ROJ-MAN 7: Other, specify						**0**

SGD ACTIVITIES CONDUCTed IN THE ROLE OF A (Pair)/DEVELOPER							
Preliminary Activities							
PR- 1: Review and agree on the overall or part of the project plan.			30				**30**
PR-2: Revise and ensure that the technology to be used is tested and understood.							**0**
PR-3: Revise and understand any appropriate internal (organizational) and external standard(s).							**0**
PR-4: Learn/relearn the organizational implementation and unit (developer) testing way of working.							**0**
PR-5: Review and revise your personal implementation and unit (developer) testing way of working.							**0**
PR-6: Other, specify							**0**
Planning Activities of a Developer Role							
PL-1: Review the requirement(s) for the unit(s) to be developed.							**0**
PL-2: Prepare (make) and/or review the design specification(s) for the unit(s) to be developed.							**0**
PL-3: Resolve unclear questions and uncertainties.							**0**
PL-4: Determine and document your implementation and unit (developer) testing goals.							**0**
PL-5: Determine your implementation and unit (developer) testing strategy.							**0**
PL-6: Determine appropriate implementation and testing practices.							**0**
PL-7: Identify standards to be used for meeting your goals.							**0**
PL-8: Set your own personal deadlines to be met during your implementation and unit (developer) testing work.							**0**
PL-9: Estimate effort and resources required for carrying out your work.				30			**30**
PL-10: Schedule your work.							**0**
PL-11: Review your implementation and unit (developer) testing plan to ensure that it is realistic and achievable.							**0**
PL-12: Identify risks related to your plan.							**0**
PL-13: Plan for managing any identified risks.							**0**
PL-14: Other, specify							**0**
Preparatory Activities of a Developer Role							
P-1: Prepare(make) and/or review your low-level design(s) of the code to be written or changed.							**0**
P-2: Prepare (make) an impact analysis of your low-level design(s).							**0**
P-3: Determine the types of unit (developer) test cases and their order.							**0**
P-4: Create and/or revise your unit (developer) test case base.							**0**
P-5: Revise the existing unit (developer) regression test base, if relevant.							**0**
P-6: Create or modify stubs and drivers, if required.							**0**
P-7: Prepare your unit (developer) testing environment and check whether it is appropriate for you work.							**0**
P-8: Other, specify							**0**
Coding Activities of a Developer Role							
C-1: Write/rewrite your code.	140		300	120	120	300	**980**
C-2: Compile/ recompile your code as required.	120		10	30	40	80	**280**
C-3: Make notes on your compilation errors, if necessary.							**0**
C-4: Make notes on your defects							**0**
C-5: Other, specify							**0**
Unit Testing Activities of a Developer Role							
T-1: Check whether the unit (developer) test case base meets the given requirements and design.							**0**
T-2: Check whether the unit (developer) regression test base meets the given requirements and design.							**0**
T-3: Remedy requirements problems in your unit (developer) regression and/or test cases base, if any.							**0**
T-4: Perform dynamic testing by executing code.							**0**
T-5: Perform static (human) testing by reviewing your code.						45	**45**
T-6: Record/write down test results.							**0**
T-7: Other, specify							**0**
Evaluative Activities of a Developer Role							
E-1: Analyze your unit (developer) testing results.							**0**
E-2: Depending on the unit (developer) testing results, determine your next step(s).							**0**
E-3: Other, specify							**0**
Debugging Activities of a Developer Role							
D-1: Identify the source of (an) error(s).	120		30	30	60		**240**
D-2: Determine solution(s) for eliminating the sources of error(s).	60		60		60		**180**
D-3: Other, specify							**0**
Self-Assessment Activities (Document aside your self-assessment results)							
A-1: Assess your own development work.							**0**
A-2: Identify causes of your mistakes.							**0**
A-3: Identify improvement areas in your own way of working.							**0**
A-4: Other, specify							**0**
Delivery of a Developer Role							
S-2: Deliver your code.(integrating)	60			20			**80**
S-3: Other, specify							**0**
						TOTAL	**2345**